HEWLETT-PACKARD COMPANY
LOGIC SYSTEMS DIVISION

HP 64000 Logic Development System

SYSTEM RELEASE BULLETIN

Part Number: **5958-6019**
E0587

Printed: **MAY 1987**

HP STARS II

SOFTWARE RELEASE BULLETIN

Issue 87.3

MAY, 1987

This document supersedes all previously dated SSBs.

HEWLETT
PACKARD

READER COMMENT SHEET

STARS II SRB (STARS B)

Issue _____._____ DATE _____/_____/_____

We welcome your evaluation of this bulletin. Your comments and suggestions help us to improve our publications. Please use additional pages if necessary.

Is this bulletin technically accurate?	Yes [ ] No [ ]	(If no, explain under Comments, below.)
Are the concepts and wording easy to understand?	Yes [ ] No [ ]	(If no, explain under Comments, below.)
Is the format of this bulletin convenient in size, arrangement and readability?	Yes [ ] No [ ]	(If no, explain or suggest improvements under Comments, below.)

Comments:

****

Date: _____

FROM:

Name _____

Company _____

Address _____

_____

_____

STARS B
Printed in U.S.A.

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, CO.

POSTAGE WILL BE PAID BY ADDRESSEE

STARS Administration
Hewlett-Packard Company
Logic Systems Division
P.O. BOX 617
Colorado Springs, Colorado 80901-0617

FOLD FOLD

P R E F A C E

This Software Release Bulletin documents all fixes and enhancements that are incorporated in the new release identified on the cover page. The SRB is provided as a benefit of Hewlett-Packard's Account Management Support, Response Center Support, and Software Materials Subscription.

Of the five sections contained in the SRB (not including the PREFACE), only the last section which contains the detailed reports has page numbers. These are referenced by the product, report number and keyword indexes in order to direct the user to a particular area or to an individual detailed report. The five sections are described below.

SOFTWARE RELEASE CONTENTS

This section lists the product names, numbers and update/fix levels of all products contained in this release. Products that have changed, or are new are denoted with an asterisk preceding the product name.

PRODUCT INDEX

Each unique product name/number has an entry listing the page number where the detailed report for that product begins.

REPORT NUMBER INDEX

This index is a sequential list of the individual report numbers with the corresponding page number where the report can be found.

KEYWORD INDEX

This index is sorted by product name, keyword, product number (including the update/fix level) and by report number in that order. In addition to the sort items, each entry has a brief description (one line) and the page number where the detailed report can be found. Note that a given report can be listed more than once in this section if it has more than one keyword assigned to it.

DETAILED REPORTS

Each report contains all the available information relevant to the problem being corrected or the enhancement being implemented.

Software release contents

Product name		Product number	uu.ff
*64000-UX OP-ENV	300	64801S004	01.50
*6800 C		64821	01.07
*6800 C	300	64821S004	01.20
*6800 C	500	64821S001	01.60
*6800 C	VAX	64821S003	01.90
*6800 PASCAL		64811	01.20
*6800 PASCAL	300	64811S004	01.20
*6800 PASCAL	500	64811S001	01.50
*6800 PASCAL	VAX	64811S003	01.70
*68000 ASSEMB		64845	01.12
*68000 ASSEMB	300	64845S004	01.20
*68000 ASSEMB	500	64845S001	01.60
*68000 ASSEMB	VAX	64845S003	01.80
*68000 C		64819	01.10
*68000 C	300	64819S004	01.20
*68000 C	500	64819S001	01.60
*68000 C	VAX	64819S003	01.90
*68000 DQ SW ANALYZER		64341G	01.03
*68000 PASCAL		64815	01.12
*68000 PASCAL	300	64815S004	01.20
*68000 PASCAL	500	64815S001	01.50
*68000 PASCAL	VAX	64815S003	01.70
*68000 SW ANAL		64331	01.02
*68000 SW ANALYZER		64341B	02.02
*68008 SW ANAL		64337	01.02
*6801/3 EMULATION	300	64256S004	01.00
*68010 DQ SW ANALYZER		64341I	01.02
*68010 SW ANAL		64334	01.02
*68010 SW ANALYZER		64341D	02.02
*68020 ASSEMB	300	64870S004	01.00
*68020 EMUL	300	64416S004	01.00
*6809 C		64822	01.08
*6809 C	300	64822S004	01.20
*6809 C	500	64822S001	01.40
*6809 C	VAX	64822S003	01.60
*6809 PASCAL		64813	01.11
*6809 PASCAL	300	64813S004	01.20
*6809 PASCAL	500	64813S001	01.30
*6809 PASCAL	VAX	64813S003	01.40
*70208 EMUL		64297	01.00
*70216 EMUL		64296	01.00
*80186 SW ANAL		64335	02.03
*80186 SW ANALYZER		64341E	02.02
*80188 SW ANAL		64336	02.04
*80188 SW ANALYZER		64341F	01.02
*80286 INTERFACE	300	64657S004	01.00
*80286B ASSEMB		64859	01.02
*80286B ASSEMB	300	64859S004	01.10
*80286B ASSEMB	500	64859S001	01.10
*80286B ASSEMB	VAX	64859S003	01.10
*8085 B PASCAL		64825	01.04
*8085 B PASCAL	300	64825S004	01.20
*8085 B PASCAL	500	64825S001	01.50
*8085 B PASCAL	VAX	64825S003	01.70
*8085 C		64826	01.04

Software release contents

Product name		Product number	uu.ff
*8085 C	300	64826S004	01.20
*8085 C	500	64826S001	01.60
*8085 C	VAX	64826S003	01.90
*8086 SW ANAL		64332	01.03
*8086 SW ANALYZER		64341A	01.02
*8086/8 ASSEMB		64853	02.02
*8086/8 ASSEMB	300	64853S004	02.20
*8086/8 ASSEMB	500	64853S001	02.30
*8086/8 ASSEMB	VAX	64853S003	02.40
*8086/8 C		64818	03.02
*8086/8 C	300	64818S004	03.20
*8086/8 C	500	64818S001	03.30
*8086/8 C	VAX	64818S003	03.50
*8086/8 PASCAL		64814	03.02
*8086/8 PASCAL	300	64814S004	03.20
*8086/8 PASCAL	500	64814S001	03.20
*8086/8 PASCAL	VAX	64814S003	03.30
*8088 DQ SW ANALYZER		64341C	01.02
*8088 SW ANAL		64333	01.03
*8088 DQ EMUL	300	64221S004	01.00
*HOST SOFTWARE /	VAX	64882	02.00
*NETWORK TRANSFER	300	64887S004	01.00
*NETWORK TRANSFER	500	64887S001	01.00
*NETWORK TRANSFER	500	64888S001	01.00
*NETWORK TRANSFER	VAX	64887S003	01.10
*NSC800 EMULATION		64292	01.03
*OPERATING SYSTEM		64100	02.07
*RS-232 TRANSFER	VAX	64886	01.20
*TIMING ANALYZER	300	64610S004	01.00
*USER DEF ASSEMB	300	64851S004	01.20
*USER DEF ASSEMB	500	64851S001	01.60
*USER DEF ASSEMB	VAX	64851S003	01.60
*USER DEF INV ASM	300	64856S004	01.00
*UTILITIES PKG	300	64888S003	01.10
*UTILITIES PKG	300	64888S004	01.00
*Z80 EMULATION	300	64252S004	01.00
*Z80/NSC800 C		64824	01.04
*Z80/NSC800 C	300	64824S004	01.20
*Z80/NSC800 C	500	64824S001	01.60
*Z80/NSC800 C	VAX	64824S003	01.90
*Z80/NSC800PASCAL		64823	01.04
*Z80/NSC800PASCAL	300	64823S004	01.20
*Z80/NSC800PASCAL	500	64823S001	01.50
*Z80/NSC800PASCAL	VAX	64823S003	01.70
*Z8000 C		64820	01.06
*Z8000 C	300	64820S004	01.20
*Z8000 C	500	64820S001	01.60
*Z8000 C	VAX	64820S003	01.90
*Z8000 PASCAL		64816	01.12
*Z8000 PASCAL	300	64816S004	01.20
*Z8000 PASCAL	500	64816S001	01.50
*Z8000 PASCAL	VAX	64816S003	01.70
*Z8001 EMUL	300	64232S004	01.00
*Z8002 EMUL	300	64233S004	01.00

Product index

Product index

Report number index

Report number index

Report number index

Keyword index

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64821	01.04	Nested switch statements may generate infinite loop	D200040378	2
	64821	01.05	Compiler is not flagging an undefined structure.	D200059782	2
	64821	01.06	C Function returning large (>2bytes) result can't be called as procedure	D200063065	3
	64821	01.06	Illegal forward reference flagged for legally defined string.	D200066084	3
PASS 1	64821	00.56	Unsigned integers treated as signed when subtracted from pointers	D200010140	1
	64821	00.56	Functions invoked via function pointers may JSR the wrong location.	D200011379	1

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64821S004	01.00	Nested switch statements may generate infinite loop	D200051946	5
	64821S004	01.00	Compiler is not flagging an undefined structure.	D200059816	5
	64821S004	01.10	C Function returning large (>2bytes) result can't be called as procedure	D200063529	6
	64821S004	01.10	Illegal forward reference flagged for legally defined string.	D200066118	6
	64821S004	01.10	No error message for unimplemented processor name.	D200066464	7

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64821S001	01.20	Nested switch statements may generate infinite loop	D200040386	8
	64821S001	01.40	Compiler is not flagging an undefined structure.	D200059790	8
	64821S001	01.50	C Function returning large (>2bytes) result can't be called as procedure	D200063503	9
	64821S001	01.50	Illegal forward reference flagged for legally defined string.	D200066092	9
	64821S001	01.50	No error message for unimplemented processor name.	D200066449	10

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64821S003	01.20	Nested switch statements may generate infinite loop	D200040394	11
	64821S003	01.50	Compiler is not flagging an undefined structure.	D200059808	11
	64821S003	01.80	C Function returning large (>2bytes) result can't be called as procedure	D200063511	12
	64821S003	01.80	Illegal forward reference flagged for legally defined string.	D200066100	12
	64821S003	01.80	No error message for unimplemented processor name.	D200066456	13

- -8

Keyword	Product number	uu.ff	Description	Report #	page
PASS 1	64811	00.61	Functional type change for one char generates a null string.	2700002980	14
RUN-TIME LIBRARY	64811	00.61	Real library routine INVALID may not be called on invalid real number.	D200010108	14

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64845	01.00	BRA.S Code does not generate properly.	5000168872	15
	64845	01.01	The assembler does not recognize invalid logical operators.	5000163626	15
	64845	01.10	Assembler allowing illegal instructions with address reg. indirect.	5000146381	15
	64845	01.10	External labels cannot be used in the "quick" type instructions.	D200061531	16
	64845	01.10	MOVEQ instruction doesn't flag an error for illegal size appensions.	D200062190	16
	64845	01.10	Illegal size appension allowed with addr reg indirect mode of addressing	D200062208	17

Keyword index

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64845	01.11	PC with index register and offset mode of addressing causing linker err.	5000175976	15
	64845	01.11	Size appensions do not always generate the appropriate error message.	D200065573	17
	64845	01.11	The immediate mode of addressing is not supported as a source operand.	D200065599	18
	64845	01.11	EXT pseudo is not supported as stated in the Assembler reference manual.	D200066563	18
	64845	01.11	LR ERROR FLAGGED WHEN USING EXPRESSION IN PC RELATIVE+IND+OFFSET ADDRing	D200066704	18
	64845	01.11	Assembler mangles displacement [PC,Xn] instructions	D200068650	19
ENHANCEMENT	64845	01.10	Assembler generating external records for symbols which are not used.	D200063206	17
	64845	01.11	Include support for the ODD psuedo to align data on an odd boundry.	D200065581	19

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64845S004	01.00	Link_sym file contains bad data in relocatable name record.	D200059451	20
	64845S004	01.00	Compiler generates duplicate symbols	D200059477	20
	64845S004	01.00	"-v" option does not work with asm inside pmon	D200059501	20
	64845S004	01.00	External labels cannot be used in the "quick" type instructions.	D200061713	20
	64845S004	01.10	EXT pseudo is not supported as stated in the Assembler reference manual.	D200066605	21

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64845S001	01.40	External labels cannot be used in the "quick" type instructions.	D200061697	22
	64845S001	01.50	Assembler reports error if file is specified with full path name.	1650024349	22
	64845S001	01.50	EXT pseudo is not supported as stated in the Assembler reference manual.	D200066589	23
ASSEMBLER	64845S001	01.40	LR error flagged for correct offset using PC+INDEX+OFFSET mode of addr.	5000136796	22
LINKER	64845S001	01.10	"Garbage" characters appear in load address statement with linker.	D200062851	23

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64845S003	01.50	External labels cannot be used in the "quick" type instructions.	D200061705	24
	64845S003	01.70	EXT pseudo is not supported as stated in the Assembler reference manual.	D200066597	24

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64819	01.07	Defining a constant hex number typecast as a pointer may fail.	D200033324	27
	64819	01.07	Code generated for return statement inside nested if's is incorrect.	D200033555	28
	64819	01.07	Nested switch statements may generate infinite loop	D200036905	29
	64819	01.08	Pass three error when an integer is assigned to a float.	5000142331	25
	64819	01.08	Compiler is not flagging an undefined structure.	5000142448	25
	64819	01.08	Compiler loads return value in two different locatations.	D200052423	31
	64819	01.08	Sign extension done when integer type cast to an unsigned long.	D200053173	32
	64819	01.08	$INIT_ZEROES$ may affect the addressing mode used for accessing var's.	D200056002	33
	64819	01.09	Illegal forward reference flagged for legally defined string.	5000161935	26
	64819	01.09	C Function returning large (>2bytes) result can't be called as procedure	D200063420	34
	64819	01.09	Compiler aborts with too many errors in pass 1.	D200065144	34
CODE GENERATOR	64819	01.07	32 bit value is treated as 64 bit value w/o first extending.	D200032029	27
PASS 1	64819	00.56	Unsigned integers treated as signed when subtracted from pointers	D200010124	26
	64819	01.07	Cannot define a function which returns a pointer to a function.	D200033597	29

Keyword index

- -8

Keyword	Product number	uu.ff	Description	Report #	page
PASS 1	64819	01.07	Wrong value calculated when scientific notation is used.	D200037358	30
PASS 3	64819	01.08	Incrementing structure member results in incomplete code generation.	D200055921	32

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64819S004	01.00	Defining a constant hex number typecast as a pointer may fail.	D200051458	35
	64819S004	01.00	Nested switch statements may generate infinite loop	D200051920	36
	64819S004	01.00	Line # labels emitted for #included files confuse analyzers	D200059493	36
	64819S004	01.00	Compiler is not flagging an undefined structure.	D200059733	37
	64819S004	01.10	C Function returning large (>2bytes) result can't be called as procedure	D200063453	37
	64819S004	01.10	Byte parameters are pushed onto the stack incorrectly.	D200064634	37
	64819S004	01.10	Illegal forward reference flagged for legally defined string.	D200066035	38
	64819S004	01.10	No error message for unimplemented processor name.	D200066415	39

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64819S001	01.10	Defining a constant hex number typecast as a pointer may fail.	D200033530	40
	64819S001	01.20	Nested switch statements may generate infinite loop	D200040329	41
	64819S001	01.40	Compiler is not flagging an undefined structure.	D200059717	42
	64819S001	01.50	List file contains control characters in a specific case.	1650017491	40
	64819S001	01.50	C Function returning large (>2bytes) result can't be called as procedure	D200063438	42
	64819S001	01.50	Byte parameters are pushed onto the stack incorrectly.	D200064618	43
	64819S001	01.50	Illegal forward reference flagged for legally defined string.	D200066019	43
	64819S001	01.50	No error message for unimplemented processor name.	D200066399	44

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64819S003	01.00	Error message are not consistient.	5000141127	45
	64819S003	01.20	Defining a constant hex number typecast as a pointer may fail.	D200033548	45
	64819S003	01.20	Nested switch statements may generate infinite loop	D200040337	46
	64819S003	01.50	Compiler is not flagging an undefined structure.	D200059725	47
	64819S003	01.80	Listing file for submitted programs is incomplete.	1650019109	45
	64819S003	01.80	C Function returning large (>2bytes) result can't be called as procedure	D200063446	48
	64819S003	01.80	Byte parameters are pushed onto the stack incorrectly.	D200064626	48
	64819S003	01.80	Illegal forward reference flagged for legally defined string.	D200066027	48
	64819S003	01.80	No error message for unimplemented processor name.	D200066407	49

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64822	01.05	Nested switch statements may generate infinite loop	D200040402	51
	64822	01.06	Compiler is not flagging an undefined structure.	D200059824	52
	64822	01.07	Return value of function call is being stored at loc. EMPTYSET.	1650019406	50
	64822	01.07	C Function returning large (>2bytes) result can't be called as procedure	D200063537	52
	64822	01.07	Illegal forward reference flagged for legally defined string.	D200066126	53
PASS 1	64822	00.06	Unsigned integers treated as signed when subtracted from pointers	D200010157	50
	64822	00.56	Functions invoked via function pointers may JSR the wrong location	D200011387	51

Keyword index

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64822S004	01.00	Nested switch statements may generate infinite loop	D200051953	54
	64822S004	01.00	Compiler is not flagging an undefined structure.	D200059857	54
	64822S004	01.10	C Function returning large (>2bytes) result can't be called as procedure	D200063560	55
	64822S004	01.10	Illegal forward reference flagged for legally defined string.	D200066159	55
	64822S004	01.10	No error message for unimplemented processor name.	D200066498	56

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64822S001	01.20	Compiler is not flagging an undefined structure.	D200059832	57
	64822S001	01.30	C Function returning large (>2bytes) result can't be called as procedure	D200063545	57
	64822S001	01.30	Illegal forward reference flagged for legally defined string.	D200066134	57
	64822S001	01.30	No error message for unimplemented processor name.	D200066472	58

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64822S003	01.20	Compiler is not flagging an undefined structure.	D200059840	59
	64822S003	01.50	C Function returning large (>2bytes) result can't be called as procedure	D200063552	59
	64822S003	01.50	Illegal forward reference flagged for legally defined string.	D200066142	59
	64822S003	01.50	No error message for unimplemented processor name.	D200066480	60

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64813	01.08	The library routine called DISPOSE does not generate correct code	5000124065	61
VARIANT RECORDS	64813	01.08	Variant records may not work.	5000098343	61

- -8

Keyword	Product number	uu.ff	Description	Report #	page
VARIANT RECORDS	64813S004	01.00	Variant records may not work.	D200051573	63

- -8

Keyword	Product number	uu.ff	Description	Report #	page
VARIANT RECORDS	64813S001	01.00	Variant records may not work.	D200036434	64

- -8

Keyword	Product number	uu.ff	Description	Report #	page
PASS 3	64813S003	00.00	Offset to parameters is incorrect in nested procedure.	1650007237	65
VARIANT RECORDS	64813S003	01.00	Variant records may not work.	D200036442	66

Keyword index

- -0

Keyword	Product number	uu.ff	Description	Report #	page
ENHANCEMENT	64859	01.00	Seperate linker outputs by adding several blank lines at the start	D200067579	68
	64859	01.00	Change the linker to only accept 80286B link_sym files	D200067595	68
	64859	01.00	File with unsupported processor name should be specified in error msg	D200067603	68
	64859	01.00	Warning message should be generated when aliasing an alias	D200067611	68
LINKER	64859	01.00	Error flag not set when file required by link is missing	D200067561	68

- -0

Keyword	Product number	uu.ff	Description	Report #	page
CODE GENERATOR	64859S004	01.00	FSTSW/FNSTSW function incorrectly with two-byte memory operand	D200055426	70
	64859S004	01.00	FSTENV instruction generates object code without required wait instr	D200055459	70
	64859S004	01.00	Obj. code generated for arithmetic instr. are incorrect.	D200055483	70

- -0

Keyword	Product number	uu.ff	Description	Report #	page
CODE GENERATOR	64859S001	01.00	FSTSW/FNSTSW function incorrectly with two-byte memory operand	D200055400	72
	64859S001	01.00	FSTENV instruction generates object code without required wait instr	D200055434	72
	64859S001	01.00	Obj. code generated for arithmetic instr. are incorrect.	D200055467	72

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64859S003	01.00	Build files generated on the VAX will not work with the 286 linker	D200067546	75
CODE GENERATOR	64859S003	01.00	FSTSW/FNSTSW function incorrectly with two-byte memory operand	D200055418	74
	64859S003	01.00	FSTENV instruction generates object code without required wait instr	D200055442	74
	64859S003	01.00	Obj. code generated for arithmetic instr. are incorrect.	D200055475	74

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64825	01.01	Compiler does not generate cross reference table.	D200020081	77
	64825	01.03	Error #1009 using byte-sized ORG'ed variables in FOR loops	D200064329	79
	64825	01.03	32-bit unsigned divide and modulus may fail	D200064428	79
	64825	01.03	Library routine REAL_ROUND may fail.	D200064493	80
	64825	01.03	DEBUG byte division and modulus may incorrectly report division by zero	D200064568	80
	64825	01.03	Set comparisons with the empty set may fail	D200064956	81
	64825	01.03	Assignment of constant string of length 1 to string variable may fail.	D200065326	81
CODE GENERATOR	64825	01.01	Compiler generates incorrect code (assignment to record variable).	D200016295	77
	64825	01.01	Incorrect code generated for adding one char to another.	D200040121	78
IF	64825	01.01	IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}	D200036863	78
PASS 2	64825	01.01	REBOOT DURING PASS 2	D200041137	79
POINTERS	64825	01.01	Variables of type pointer may not be incremented correctly.	D200029793	77

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64825S004	01.10	Error #1009 using byte-sized ORG'ed variables in FOR loops	D200064352	85
	64825S004	01.10	32-bit unsigned divide and modulus may fail	D200064451	85

Keyword index

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64825S004	01.10	Library routine REAL_ROUND may fail.	D200064527	86
	64825S004	01.10	DEBUG byte division and modulus may incorrectly report division by zero	D200064592	86
	64825S004	01.10	Set comparisons with the empty set may fail	D200064980	86
	64825S004	01.10	Assignment of constant string of length 1 to string variable may fail.	D200065359	87
CODE GENERATOR	64825S004	01.00	Compiler generates incorrect code (assignment to record variable).	D200050203	83
	64825S004	01.00	Incorrect code generated for adding one char to another.	D200051862	84
IF	64825S004	01.00	IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}	D200051607	84
POINTERS	64825S004	01.00	Variables of type pointer may not be incremented correctly.	D200051094	83

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64825S001	01.40	Error #1009 using byte-sized ORG'ed variables in FOR loops	D200064337	89
	64825S001	01.40	32-bit unsigned divide and modulus may fail	D200064436	90
	64825S001	01.40	Library routine REAL_ROUND may fail.	D200064501	90
	64825S001	01.40	DEBUG byte division and modulus may incorrectly report division by zero	D200064576	90
	64825S001	01.40	Set comparisons with the empty set may fail	D200064964	91
	64825S001	01.40	Assignment of constant string of length 1 to string variable may fail.	D200065334	91
CODE GENERATOR	64825S001	01.10	Compiler generates incorrect code (assignment to record variable).	D200016311	88
	64825S001	01.20	Incorrect code generated for adding one char to another.	D200040139	89
IF	64825S001	01.20	IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}	D200036848	88
POINTERS	64825S001	01.10	Variables of type pointer may not be incremented correctly.	D200029801	88

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64825S003	01.50	Using char and int. in control loop causes incorrect code to be gen'ed.	D200058677	94
	64825S003	01.50	$Range ON$ causes incorrect code to be generated for a test operation.	D200059659	95
	64825S003	01.50	Incorrect data offsets in listing file.	D200060244	95
	64825S003	01.60	functional type change of a constant into multi-byte structure gen's err	D200063925	96
	64825S003	01.60	Error #1009 using byte-sized ORG'ed variables in FOR loops	D200064345	97
	64825S003	01.60	32-bit unsigned divide and modulus may fail	D200064444	97
	64825S003	01.60	Library routine REAL_ROUND may fail.	D200064519	98
	64825S003	01.60	DEBUG byte division and modulus may incorrectly report division by zero	D200064584	98
	64825S003	01.60	Set comparisons with the empty set may fail	D200064972	98
	64825S003	01.60	Assignment of constant string of length 1 to string variable may fail.	D200065342	99
	64825S003	01.60	.A, .R, and listing files should reside in directory compile is executed.	D200067017	100
	64825S003	01.60	Assignment of unsigned_8 variables to expression always assigns zero.	D200067447	100
CODE GENERATOR	64825S003	01.10	Compiler generates incorrect code (assignment to record variable).	D200016303	93
	64825S003	01.20	Incorrect code generated for adding one char to another.	D200040147	94
IF	64825S003	01.20	IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}	D200036855	93
POINTERS	64825S003	01.20	Variables of type pointer may not be incremented correctly.	D200029819	93

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64826	01.01	IF statements involving return values and address calculations may fail.	D200037622	102
	64826	01.01	Nested switch statements may generate infinite loop	D200040444	103
	64826	01.02	Compiler is not flagging an undefined structure.	D200059907	104
	64826	01.02	Incorrect code generated when function parameter is post incremented.	D200062828	104
	64826	01.03	C Function returning large (>2bytes) result can't be called as procedure	D200063610	105

Keyword index

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64826	01.03	Funct calls via pointers with parms cause subsequent stack ref errors	D200064881	105
	64826	01.03	Illegal forward reference flagged for legally defined string.	D200066209	106
CODE GENERATOR	64826	01.01	Assigning a ptr. after its post incr/decr. gives incorrect value.	D200025742	102
	64826	01.03	Character isn't converted to int before calculations use it	D200063297	105
PASS 1	64826	01.00	Functions invoked via function pointers may JSR the wrong location.	D200011262	101
	64826	01.00	Unsigned integers treated as signed when subtracted from pointers.	D200011346	101

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64826S004	01.00	Nested switch statements may generate infinite loop	D200051979	108
	64826S004	01.00	Compiler is not flagging an undefined structure.	D200059931	108
	64826S004	01.10	C Function returning large (>2bytes) result can't be called as procedure	D200063644	110
	64826S004	01.10	Funct calls via pointers with parms cause subsequent stack ref errors	D200064915	110
	64826S004	01.10	Illegal forward reference flagged for legally defined string.	D200066233	110
	64826S004	01.10	No error message for unimplemented processor name.	D200066555	111
CODE GENERATOR	64826S004	01.10	Character isn't converted to int before calculations use it	D200063321	109

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64826S001	01.05	Number of errors listed at bottom of the listing is incorrect.	5000179028	112
	64826S001	01.20	Nested switch statements may generate infinite loop	D200040451	113
	64826S001	01.20	IF statements involving return values and address calculations may fail.	D200042085	114
	64826S001	01.40	Compiler is not flagging an undefined structure.	D200059915	114
	64826S001	01.50	C Function returning large (>2bytes) result can't be called as procedure	D200063628	115
	64826S001	01.50	Funct calls via pointers with parms cause subsequent stack ref errors	D200064899	116
	64826S001	01.50	Illegal forward reference flagged for legally defined string.	D200066217	116
	64826S001	01.50	No error message for unimplemented processor name.	D200066530	117
CODE GENERATOR	64826S001	01.10	Assigning a ptr. after its post incr/decr. gives incorrect value.	D200025759	112
	64826S001	01.50	Character isn't converted to int before calculations use it	D200063305	115

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64826S003	01.20	Nested switch statements may generate infinite loop	D200040469	118
	64826S003	01.20	IF statements involving return values and address calculations may fail.	D200042093	119
	64826S003	01.60	Compiler is not flagging an undefined structure.	D200059923	120
	64826S003	01.80	C Function returning large (>2bytes) result can't be called as procedure	D200063636	121
	64826S003	01.80	Funct calls via pointers with parms cause subsequent stack ref errors	D200064907	121
	64826S003	01.80	Illegal forward reference flagged for legally defined string.	D200066225	121
	64826S003	01.80	No error message for unimplemented processor name.	D200066548	122
CODE GENERATOR	64826S003	01.10	Assigning a ptr. after its post incr/decr. gives incorrect value.	D200025767	118
	64826S003	01.80	Character isn't converted to int before calculations use it	D200063313	120

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64853	02.00	Corrupt file generated by assem. when large # of files are link. w/xref	5000136226	123
	64853	02.00	STACKSEG pseudo op does not allocate space correctly.	D200033563	125

Keyword index

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64853	02.00	Expression type errors occur for legal INC instructions.	D200042242	125
	64853	02.00	Macro called with more parameters than declared generates error.	D200043885	126
	64853	02.01	Assermbler does not flag LR error when short jump > +/- 127 bytes	5000152090	123
	64853	02.01	OLD_8087 directive is ignored after the use of DQ pseudo	5000154542	124
	64853	02.01	FMUL ST[3],ST[5] does not flag error	5000161836	124
CODE GENERATOR	64853	02.00	Index addressing in MOV statement creates incorrect code	5000136093	123
LINKER	64853	00.08	"Total # of bytes loaded" is incorrect if segment boundary is crossed.	D200005116	125

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64853S004	02.00	Expression type errors occur for legal INC instructions.	D200052100	127

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64853S001	02.00	Expression type errors occur for legal INC instructions.	D200042556	128

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64853S003	02.00	Expression type errors occur for legal INC instructions.	D200042564	129

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64818	02.00	Dereferencing a structue is not working properly.	5000108969	130
	64818	02.00	Nested switch statements may generate infinite loop	D200040295	133
	64818	03.00	Compiler is not flagging an undefined structure.	D200059675	133
	64818	03.01	C Function returning large (>2bytes) result can't be called as procedure	D200063388	134
	64818	03.01	Illegal forward reference flagged for legally defined string.	D200065979	134
CODE GENERATOR	64818	00.56	Error #1006 generated when incorrect value returned from a function	D200007831	131
	64818	02.00	AX not loaded with constant prior to using it to calculate expression	5000135913	130
	64818	02.00	The compiler generates incorrect code for floating point constants	5000160770	130
PASS 1	64818	00.56	Unsigned integers treated as signed when subtracted from pointers.	D200010116	132
	64818	00.56	Functions invoked via function pointers may JSR the wrong location.	D200011395	132

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64818S004	03.00	With $POINTER_SIZE 32$ assigning an address + a sizeof in 1 line fails.	D200049973	136
	64818S004	03.00	Nested switch statements may generate infinite loop	D200051912	136
	64818S004	03.00	Compiler is not flagging an undefined structure.	D200059709	137
	64818S004	03.10	C Function returning large (>2bytes) result can't be called as procedure	D200063412	137
	64818S004	03.10	Illegal forward reference flagged for legally defined string.	D200066001	137
	64818S004	03.10	No error message for unimplemented processor name.	D200066381	138

Keyword index

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64818S001	02.01	Nested switch statements may generate infinite loop	D200040303	142
	64818S001	02.01	File will not compile on the 9000/500.	D200045559	143
	64818S001	03.00	Both operands of expression loaded into AX when calculating array index	5000149773	139
	64818S001	03.10	Compiler is not flagging an undefined structure.	D200059683	143
	64818S001	03.20	C Function returning large (>2bytes) result can't be called as procedure	D200063396	144
	64818S001	03.20	Illegal forward reference flagged for legally defined string.	D200065987	144
	64818S001	03.20	No error message for unimplemented processor name.	D200066365	145
CODE GENERATOR	64818S001	03.00	ES registeris overwritten when loading a ptr. w/ addr.of a structure	5000152108	140
	64818S001	03.00	Compiler generates MOVSB without init. ES - POINTER -> member = VAR;	5000154245	141

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64818S003	02.00	Data space cannot exceed 32K.	5000114645	146
	64818S003	02.00	Nested switch statements may generate infinite loop	D200040311	147
	64818S003	03.10	Compiler aborts when incorrectly passing address of array as funct. para	5000129817	146
	64818S003	03.10	Compiler is not flagging an undefined structure.	D200059691	147
	64818S003	03.40	C Function returning large (>2bytes) result can't be called as procedure	D200063404	148
	64818S003	03.40	Illegal forward reference flagged for legally defined string.	D200065995	148
	64818S003	03.40	No error message for unimplemented processor name.	D200066373	149
CODE GENERATOR	64818S003	02.01	float/double vars. in a subroutine uses MOVESB without init. ES	5000128959	146

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64814	02.00	Incorrect code generated for assignment statement.	D200030775	150
	64814	02.01	Program reboots or aborts with too many errors (64000 / host).	D200037325	150
CODE GENERATOR	64814	03.00	$SEPARATE_CONST OFF$ USED WITH REAL # CONSTS. GENERATES POP CS/PUSH CS.	D200055335	150
	64814	03.01	Record members' addresses are calcul. incorrectly inside the WITH stmnt	D200063990	151
	64814	03.01	SHORT JMP generated instead of NEAR JMP when jumping > 32K	D200065078	152
PASS 3	64814	03.01	SHORT JMP generated instead of NEAR JMP when jumping > 32K	D200065078	152

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64814S004	03.00	Incorrect code generated for assignment statement.	D200051219	153
	64814S004	03.00	Program reboots or aborts with too many errors (64000 / host).	D200051797	153
CODE GENERATOR	64814S004	03.00	$SEPARATE_CONST OFF$ USED WITH REAL # CONSTS. GENERATES POP CS/PUSH CS.	D200055517	153
	64814S004	03.10	Record members' addresses are calcul. incorrectly inside the WITH stmnt	D200064097	154
PASS 2	64814S004	03.00	Too many errors, pass2: 80186 (PROCEDURE, WITH statement).	D200050245	153

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64814S001	02.00	Incorrect code generated for assignment statement.	D200030783	156
	64814S001	02.00	Program reboots or aborts with too many errors (64000 / host).	D200037333	156
CODE GENERATOR	64814S001	03.00	$SEPARATE_CONST OFF$ USED WITH REAL # CONSTS. GENERATES POP CS/PUSH CS.	D200055491	156
	64814S001	03.10	Record members' addresses are calcul. incorrectly inside the WITH stmnt	D200064071	157
PASS 2	64814S001	01.10	Too many errors, pass2: 80186 (PROCEDURE, WITH statement).	D200025908	156

Keyword index

- -O

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64814S003	02.00	Incorrect code generated for assignment statement.	D200030791	159
	64814S003	02.00	Program reboots or aborts with too many errors (64000 / host).	D200037341	159
CODE GENERATOR	64814S003	03.00	$SEPARATE_CONST OFF$ USED WITH REAL # CONSTS. GENERATES POP CS/PUSH CS.	D200055509	159
	64814S003	03.20	Record members' addresses are calcul. incorrectly inside the WITH stmnt	D200064089	160
PASS 2	64814S003	01.10	Too many errors, pass 2: 80186 (PROCEDURE, WITH statement).	D200025916	159

- -O

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64882	01.10	File name conversion (transfer) is inconsistent with COMP and ASM.	D200021790	162
	64882	01.20	Transfer may not function across VAX-cluster.	D200046102	162
	64882	01.70	Misspellings in HPINSTALL.COM can cause %F-ERROR.	D200065680	163
	64882	01.70	HSL will not start with most 64000 printers (introduced in 1.7)	D200067512	163
HIGH SPEED LINK	64882	01.20	Initializing the HSL may require more than one shift/reset on the 64000.	D200047951	162
	64882	01.20	HSLSTOP doesn't work if MAPBUS is pending.	D200048017	162
	64882	01.20	IBDRIVER conficts with existing driver on the system.	D200048041	164
MAPBUS	64882	01.60	Define MAPBUS as a verb in HPTABLES.CLD instead of a symbol in HPSETUP.	D200055012	164
RCMAIN	64882	01.60	RCMAIN/VERBOSE not described in the HELP file.	D200054775	163
TRANSFER	64882	01.20	Insufficient examples in the HELP entry.	D200045088	162
	64882	01.20	TRANSFER does not timeout.	D200047845	163
	64882	01.50	CLUSTER-CLUSTER transfers don't work.	D200048140	163

- -S

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64292	01.02	Incorrect Inverse Assembly with State when restart active	D200060285	165
	64292	01.02	NSC800 cannot access the last 256 byte block of user memory.	D200067470	165
	64292	01.02	"modify register PC" immediately after "load <absolute_file>" fails	D200067488	166

- -P

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64100	02.06	MAIN Assemb stops table interpretation for expressions delinited by "."	D200072199	167

- -S

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64851S004	01.00	Problem with timemark in hosted assemblers.	D200061614	168
	64851S004	01.00	EQU pseudo with OLLH for an operand may halt assembly.	D200062653	168
	64851S004	01.10	Assembler trys to assemble .A files.	D200065011	168
	64851S004	01.10	Assembler aborts when full path name is specified.	D200067454	168
LINKER	64851S004	01.10	Linker does not correctly handle "NO LOAD" files.	D200063248	168

- -S

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64851S001	01.30	ASM is unable to assemble a file accessed across lan via a netunam.	D200055384	170
	64851S001	01.40	Comma at the end of a HEX pseudo statement causes the assembler to hang.	5000149211	170
	64851S001	01.40	Problem with timemark in hosted assemblers.	D200061598	170

Keyword index

- -S

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64851S001	01.50	Assembler trys to assemble .A files.	D200064840	171
	64851S001	01.50	Assembler aborts when full path name is specified.	D200067439	171
LINKER	64851S001	01.50	Linker does not correctly handle "NO LOAD" files.	D200063230	170

- -S

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64851S003	01.50	EQU pseudo with OLLH for an operand may halt assembly.	D200062646	172
	64851S003	01.50	Assembler trys to assemble .A files.	D200065003	172
LINKER	64851S003	01.04	Linker does not correctly handle "NO LOAD" files.	5000143370	172
	64851S003	01.50	Displacement > 32K error being flagged when it should not be.	D200060830	172

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64824	00.00	Changes to pointers to unions does not work properly in C language.	2700003921	174
	64824	00.00	Certain single argument Rvalues will not compile correctly.	2700003939	174
	64824	00.00	Library routine REAL_SUB modifies DE register pair.	2700004093	174
	64824	01.01	Incorredt or NO listing file produced if fatal pass 2 errors (#10xx)	D200034918	178
	64824	01.01	DIF AND WRONG CODE PRODUCED IF ARRAY ELEMENT ASSIGNED RESULT OF INDIRECT	D200037697	179
	64824	01.01	Nested switch statements may generate infinite loop	D200040410	179
	64824	01.02	Compiler is not flagging an undefined structure.	D200059865	180
	64824	01.03	Funct calls via pointers with parms cause subsequent stack ref errors	D200063032	181
	64824	01.03	C Function returning large (>2bytes) result can't be called as procedure	D200063578	181
	64824	01.03	Illegal forward reference flagged for legally defined string.	D200066167	181
CODE GENERATOR	64824	01.00	Assigning a ptr. after its post incr/decr. gives incorrect value.	D200013300	176
	64824	01.01	Registers used by Zbshift loaded incorrectly after structure reference.	2700005603	174
	64824	01.01	Operating on parm. in function call generates incorrect code.	D200022301	178
	64824	01.01	Pointer addressing wrong location after it has been updated.	D200022624	178
	64824	01.03	Character isn't converted to int before calculations use it	5000139204	175
PASS 1	64824	01.00	Functions invoded via function pointers may JSR the wrong location.	D200011148	175
	64824	01.00	Unsigned integers treated as signed when subtracted from pointers.	D200011221	176
PASS 2	64824	01.01	Pass 2 error #1006 in if construct when subtracting a const. from a var.	D200015966	177

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64824S004	01.00	Nested switch statements may generate infinite loop	D200051961	183
	64824S004	01.00	Compiler is not flagging an undefined structure.	D200059899	183
	64824S004	01.10	C Function returning large (>2bytes) result can't be called as procedure	D200063602	185
	64824S004	01.10	Funct calls via pointers with parms cause subsequent stack ref errors	D200064873	185
	64824S004	01.10	Illegal forward reference flagged for legally defined string.	D200066191	185
	64824S004	01.10	No error message for unimplemented processor name.	D200066522	186
CODE GENERATOR	64824S004	01.10	Character isn't converted to int before calculations use it	D200063289	184

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64824S001	01.20	Nested switch statements may generate infinite loop	D200040428	188
	64824S001	01.40	Compiler is not flagging an undefined structure.	D200059873	188

Keyword index

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64824S001	01.50	C Function returning large (>2bytes) result can't be called as procedure	D200063586	190
	64824S001	01.50	Funct calls via pointers with parms cause subsequent stack ref errors	D200064857	190
	64824S001	01.50	Illegal forward reference flagged for legally defined string.	D200066175	190
	64824S001	01.50	No error message for unimplemented processor name.	D200066506	191
CODE GENERATOR	64824S001	01.10	Assigning a ptr. after its post incr/decr. gives incorrect value.	D200025726	187
	64824S001	01.50	Character isn't converted to int before calculations use it	D200063263	189
PASS 2	64824S001	01.00	Pass 2 Error #1006 when subracting a const. from a var. in an if constr.	D200015982	187

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64824S003	01.20	Nested switch statements may generate infinite loop	D200040436	193
	64824S003	01.50	Compiler is not flagging an undefined structure.	D200059881	193
	64824S003	01.80	C Function returning large (>2bytes) result can't be called as procedure	D200063594	195
	64824S003	01.80	Funct calls via pointers with parms cause subsequent stack ref errors	D200064865	195
	64824S003	01.80	Illegal forward reference flagged for legally defined string.	D200066183	195
	64824S003	01.80	No error message for unimplemented processor name.	D200066514	196
CODE GENERATOR	64824S003	01.10	Assigning a ptr. after its post incr/decr. gives incorrect value.	D200025734	192
	64824S003	01.80	Character isn't converted to int before calculations use it	D200063271	194
PASS 2	64824S003	01.00	Pass 2 Error #1006 when subtracting a const. from a var. in an if constr	D200015974	192

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64823	01.01	Compiler does not generate cross reference table.	D200020099	199
	64823	01.02	Error #1006 when accessing an element of a two-dimensional array.	5000146407	197
	64823	01.02	Assignment to multi-dimensional array causes error 1006.	5000157180	198
	64823	01.03	Error #1009 using byte-sized ORG'ed variables in FOR loops	D200062976	200
	64823	01.03	32-bit unsigned divide and modulus may fail	D200062984	200
	64823	01.03	Library routine REAL_ROUND may fail.	D200062992	201
	64823	01.03	Set comparisons with the empty set may fail	D200063008	201
	64823	01.03	DEBUG byte division and modulus may incorrectly report division by zero	D200063016	202
	64823	01.03	Assignment of constant string of length 1 to string variable may fail.	D200065292	202
CODE GENERATOR	64823	01.01	Incorrect code generated for adding one char to another.	5000105841	197
ENHANCEMENT	64823	01.01	More accurate error message when wrong parm type is passed to STRWRITE.	5000136986	203
IF	64823	01.01	IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}	5000099176	197
PASS 2	64823	01.01	REBOOT DURING PASS 2 - related to position of variable declarations.	D200037507	200
PASS 3	64823	01.03	Error 1113 generated during pass 3 when 23rd label is encountered.	D200063214	202
POINTERS	64823	01.01	Variables of type pointer may not be incremented correctly.	D200029744	199

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64823S004	01.10	Error #1009 using byte-sized ORG'ed variables in FOR loops	D200064311	205
	64823S004	01.10	32-bit unsigned divide and modulus may fail	D200064410	206
	64823S004	01.10	Library routine REAL_ROUND may fail.	D200064485	206
	64823S004	01.10	DEBUG byte division and modulus may incorrectly report division by zero	D200064550	206
	64823S004	01.10	Set comparisons with the empty set may fail	D200064949	207
	64823S004	01.10	Assignment of constant string of length 1 to string variable may fail.	D200065318	207
CODE GENERATOR	64823S004	01.00	Incorrect code generated for adding one char to another.	D200051854	205
IF	64823S004	01.00	IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}	D200051599	205

Keyword index

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64823S001	01.40	Error #1009 using byte-sized ORG'ed variables in FOR loops	D200064295	209
	64823S001	01.40	32-bit unsigned divide and modulus may fail	D200064394	209
	64823S001	01.40	Library routine REAL_ROUND may fail.	D200064469	210
	64823S001	01.40	DEBUG byte division and modulus may incorrectly report division by zero	D200064535	210
	64823S001	01.40	Set comparisons with the empty set may fail	D200064923	211
	64823S001	01.40	Assignment of constant string of length 1 to string variable may fail.	D200065284	211
CODE GENERATOR	64823S001	01.20	Incorrect code generated for adding one char to another.	D200040105	209
IF	64823S001	01.20	IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}	D200036673	208
POINTERS	64823S001	01.10	Variables of type pointer may not be incremented correctly.	D200029777	208

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64823S003	01.60	Error #1009 using byte-sized ORG'ed variables in FOR loops	D200064303	214
	64823S003	01.60	32-bit unsigned divide and modulus may fail	D200064402	214
	64823S003	01.60	Library routine REAL_ROUND may fail.	D200064477	215
	64823S003	01.60	DEBUG byte division and modulus may incorrectly report division by zero	D200064543	215
	64823S003	01.60	Set comparisons with the empty set may fail	D200064931	216
	64823S003	01.60	Assignment of constant string of length 1 to string variable may fail.	D200065300	216
CODE GENERATOR	64823S003	01.20	Incorrect code generated for adding one char to another.	D200040113	214
IF	64823S003	01.20	IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}	D200036681	213
POINTERS	64823S003	01.20	Variables of type pointer may not be incremented correctly.	D200029785	213

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64820	01.03	RANGE_ON	D200014498	219
	64820	01.03	Nested switch statements may generate infinite loop	D200040345	219
	64820	01.04	Compiler is not flagging an undefined structure.	D200059741	220
	64820	01.05	C Function returning large (>2bytes) result can't be called as procedure	D200063481	220
	64820	01.05	Illegal forward reference flagged for legally defined string.	D200066043	220
PASS 1	64820	00.56	Unsigned integers treated as signed when subtracted from pointers	D200010132	218
	64820	00.56	Functions invoked via function pointers may JSR the wrong location.	D200011403	218

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64820S004	01.00	Nested switch statements may generate infinite loop	D200051938	222
	64820S004	01.00	Compiler is not flagging an undefined structure.	D200059774	222
	64820S004	01.10	C Function returning large (>2bytes) result can't be called as procedure	D200063495	223
	64820S004	01.10	Illegal forward reference flagged for legally defined string.	D200066076	223
	64820S004	01.10	No error message for unimplemented processor name.	D200066431	224

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64820S001	01.20	Nested switch statements may generate infinite loop	D200040352	225
	64820S001	01.40	Compiler is not flagging an undefined structure.	D200059758	225
	64820S001	01.50	C Function returning large (>2bytes) result can't be called as procedure	D200063479	226
	64820S001	01.50	Illegal forward reference flagged for legally defined string.	D200066050	226

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64820S001	01.50	No error message for unimplemented processor name.	D200066423	227

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64820S003	01.20	Nested switch statements may generate infinite loop	D200040360	228
	64820S003	01.50	Compiler is not flagging an undefined structure.	D200059766	229
	64820S003	01.80	No error message for unimplemented processor name.	1650018804	228
	64820S003	01.80	C Function returning large (>2bytes) result can't be called as procedure	D200063487	229
	64820S003	01.80	Illegal forward reference flagged for legally defined string.	D200066068	229

Number: D200010140 Product: 6800 C 64821 00.56

Keywords: PASS 1

One-line description:
Unsigned integers treated as signed when subtracted from pointers

Problem:
When an unsigned short or integer is used as an offset to a pointer, the unsigned will be treated as a signed when doing pointer calculations. Offsets large enough to set the sign bit will be interpreted as a negative offset when the offset is subtracted from a pointer. The following code exibits the problem if offset is greater than 32767 dec.

```
unsigned offset;
struct { int a,b,c;
       } *ptr;
unsigned long x;

main ()
{
  x = ptr - offset;  /* The compiler will generate code negating   */
}                    /* offset for the "-" operation.              */
```

Temporary solution:
Cast the offset in the expression as the next larger integer.
ie. x = ptr - (unsigned long)offset;

Signed off 04/29/87 in release 101.07

Number: D200011379 Product: 6800 C 64821 00.56

Keywords: PASS 1

One-line description:
Functions invoked via function pointers may JSR the wrong location.

Problem:
When the typedef statement is used to define pointers to functions, and this pointer type is used in a cast of a variable array to invoke code stored in that array, program execution may transfer to the wrong location. For example, in the following code the simple call to code_array fails while the call and assignment to p works correctly:

```
typedef int(*PFI)();  /* PFI a pointer to int functions */
int code_array[100];  /* array contains code */
PFI p;                /* p a pointer of type PFI */

pfibug()
{
  (*((PFI) code_array))();  /* fails in JSR to code_array */
  (*(p=(PFI)code_array))(); /* assignment and JSR successful */
}
```

Temporary solution:
Set up a dummy variable and perform an assignment to it when doing this type of operation.

Signed off 04/29/87 in release 101.07

Number: D200040378 Product: 6800 C 64821 01.04

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
       int  c;
            switch(c){
                 case1:         break;
                 default:       switch(c){
                                   case 2: break;
                                }
                                break;
            }
}
```

Signed off 04/29/87 in release 101.07

Number: D200059782 Product: 6800 C 64821 01.05

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"
```

```
main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 101.07

Number: D200063065 Product: 6800 C 64821 01.06

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 101.07

Number: D200066084 Product: 6800 C 64821 01.06

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:
"C"
"processor"

```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

"C"
"processor"

```
char  string[]  =  "do it this way";

main()
```



```
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 101.07

Number: D200051946 Product: 6800 C 300 64821S004 01.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"processor name"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"processor name"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 401.20

Number: D200059816 Product: 6800 C 300 64821S004 01.00

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {
```



```
int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 401.20

Number: D200063529 Product: 6800 C 300 64821S004 01.10

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 401.20

Number: D200066118 Product: 6800 C 300 64821S004 01.10

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work");
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;
```

```
  i = sizeof(string);
}
```

Signed off 04/29/87 in release 401.20

Number: D200066464 Product: 6800 C 300 64821S004 01.10

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 401.20

Number: D200040386 Product: 6800 C 500 64821S001 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 101.60

Number: D200059790 Product: 6800 C 500 64821S001 01.40

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {
```

```
int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 101.60

Number: D200063503 Product: 6800 C 500 64821S001 01.50

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 101.60

Number: D200066092 Product: 6800 C 500 64821S001 01.50

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:
"C"
"processor"

```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

"C"
"processor"

```
char  string[]  =  "do it this way";

main()
{
  int i;
```



```
  i = sizeof(string);
}
```

Signed off 04/29/87 in release 101.60

Number: D200066449 Product: 6800 C 500 64821S001 01.50

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 101.60

Number: D200040394 Product: 6800 C VAX 64821S003 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 301.90

Number: D200059808 Product: 6800 C VAX 64821S003 01.50

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {
```



```
int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 301.90

Number: D200063511 Product: 6800 C VAX 64821S003 01.80

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 301.90

Number: D200066100 Product: 6800 C VAX 64821S003 01.80

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;
```

```
  i = sizeof(string);
}
```

Signed off 04/29/87 in release 301.90

Number: D200066456 Product: 6800 C VAX 64821S003 01.80

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 301.90

Number: 2700002980 Product: 6800 PASCAL 64811 00.61

Keywords: PASS 1

One-line description:
Functional type change for one char generates a null string.

Signed off 04/29/87 in release 101.20

Number: D200010108 Product: 6800 PASCAL 64811 00.61

Keywords: RUN-TIME LIBRARY

One-line description:
Real library routine INVALID may not be called on invalid real number.

Problem:
The real number library routine INVALID may not be called when an invalid floating point number is passed as a parameter to one of the floating point routines.

Signed off 04/29/87 in release 101.20

Number: 5000146381 Product: 68000 ASSEMB 64845 01.10

One-line description:
Assembler allowing illegal instructions with address reg. indirect.

Problem:
The 68000 assembler allows the PC to be used in apparently all variations off the address register indirect mode of addressing.

"68000"

```
        MOVE.W     D0,-[PC]          ;GENS CODE OF D0,-[A0]
        MOVE.L     D0,[PC]
```

Signed off 04/29/87 in release 501.12

Number: 5000163626 Product: 68000 ASSEMB 64845 01.01

One-line description:
The assembler does not recognize invalid logical operators.

Problem:
If you use incorrect syntax for logical operators the assembler aborts the instruction, but, does not flag an error.

"68000"

```
        PROG
        MOVE.W          #10001B.or.01110B,D2      ;FLAGGED CORRECTLY

VALUE   EQU             10001B.or.01110B          ;NOT FLAGGED

LABEL   EQU             10101010B.AND.0FH.OR.30   ;NOT FLAGGED

EXAMPLE EQU             0FFH.invalid.0AH          ;NOT FLAGGED
```

Signed off 04/29/87 in release 501.12

Number: 5000168872 Product: 68000 ASSEMB 64845 01.00

One-line description:
BRA.S Code does not generate properly.

Signed off 04/29/87 in release 501.12

Number: 5000175976 Product: 68000 ASSEMB 64845 01.11

One-line description:
PC with index register and offset mode of addressing causing linker err.

Problem:
Relocatable file generated may be incorrect for certain instructions.

"68000"


```
        MOVE.L   -16[PC,D2],D1
```

This code assembles without errors, but, causes a linker error.
ERROR: Displacement > 32k.

Signed off 04/29/87 in release 501.12

Number: D200061531 Product: 68000 ASSEMB 64845 01.10

One-line description:
External labels cannot be used in the "quick" type instructions.

Problem:
You cannot use an external label as data in the "quick" type instructions. If you have two files:

file: declare

"68000"

```
        GLB     EXTLAB
EXTLAB  EQU     4
```

file: refer

"68000"

```
        EXTERNAL        EXTLAB
LABEL   EQU             7

        MOVEQ.L         #EXTLAB,D1      ;IO error is flagged
        MOVEQ.L         #LABEL,D1       ;WORKS
```

Temporary solution:
Do not use external variables in the "quick" type instructions. You can possibly get around this by "including" the symbol (via an include file) rather than declaring it external.

Signed off 04/29/87 in release 501.12

Number: D200062190 Product: 68000 ASSEMB 64845 01.10

One-line description:
MOVEQ instruction doesn't flag an error for illegal size appensions.

Problem:
Illegal size appension on the MOVEQ instruction are not flagged with a warning. If you have

"68000"

```
        MOVEQ.W     #1,D0
        MOVEQ.B     #1,D0
```

The assembler will not flag an error and generates code for a MOVEQ.L.

Signed off 04/29/87 in release 501.12

Number: D200062208 Product: 68000 ASSEM8 64845 01.10

One-line description:
Illegal size appension allowed with addr reg indirect mode of addressing

Problem:
When using the address register with displacement and index reg. mode of addressing the assembler does not flag an error for an illegal size appension on the index reg.

"68000"

```
     MOVE       8[A0,D4.8],D1    ;.B IS ILLEGAL
```

The assembler executes the instruction assuming a word appension on D4, but, fails to generate a warning or error message.

Signed off 04/29/87 in release 501.12

Number: D200063206 Product: 68000 ASSEMB 64845 01.10

Keywords: ENHANCEMENT

One-line description:
Assembler generating external records for symbols which are not used.

Problem:
If an external symbol is declared in a source file and is not used then the assembler should not generate an external record.

"processor"

```
       EXTERNAL   NOTUSED
       END
```

If you do a link, NOTUSED can be found in the XREF.

Signed off 04/29/87 in release 501.12

Number: D200065573 Product: 68000 ASSEM8 64845 01.11

One-line description:
Size appensions do not always generate the appropriate error message.

Problem:
Some instructions allow illegal size appensions while another flags an error for a legal appension. You may append the MOVEQ instruction with any size attribute and no error is reported. The correct code is generated for illegal appensions Secondly, the D8cc instruction must always be of word size. If you try to assemble D8cc.W the assembler flags an error.

Signed off 04/29/87 in release 501.12

Number: D200065599 Product: 68000 ASSEMB 64845 01.11

One-line description:
The immediate mode of addressing is not supported as a source operand.

Problem:
The BTST instruction should support the immediate data mode of addressing for its source operand. It doesn't.

Signed off 04/29/87 in release 501.12

Number: D200066563 Product: 68000 ASSEM8 64845 01.11

One-line description:
EXT pseudo is not supported as stated in the Assembler reference manual.

Problem:

The Assembler/Linker Reference Manual states that either EXT or EXTERNAL may be used when declaring an external label. The 68000 assembler only accepts EXTERNAL.

Temporary solution:
Always use EXTERNAL.

Signed off 04/29/87 in release 501.12

Number: D200066704 Product: 68000 ASSEMB 64845 01.11

One-line description:
LR ERROR FLAGGED WHEN USING EXPRESSION IN PC RELATIVE+IND+OFFSET ADDRing

Problem:
A legal range error will be flagged when using the PC relative mode of addressing with offset and index register, if, the offset is an expression.

"68000"

```
        ORG     1000H                   ;ORG PAST VECTOR TA8LE
        MOVE    LABEL+1[PC,D0],D0       ;LR ERROR FLAGGED
LA8EL   1000H
```

Temporary solution:
Do not use an expression for the offset. You can avoid this by using the index register for offseting.

"68000"

```
        ORG     1000H
        MOVE    OFFSET_VALUE,D0         ;OFFSET_VALUE IS NOT INTENDED
                                        ;TO BE A CONSTANT.

        MOVE    LA8EL[PC,D0],D1         ;D0 CONTAINS OFFSET VALUE
```

Signed off 04/29/87 in release 501.12

Number: D200068650 Product: 68000 ASSEMB 64845 01.11

One-line description:
Assembler mangles displacement [PC,Xn] instructions

Problem:
Assembler incorrectly passes the value of absolute displacements to the Linker for operands of the form: displacement [PC,A0]. This is due to their being passed as relocatable code to the linker, where they are subseuently mangled.

Signed off 04/29/87 in release 501.12

Number: D200065581 Product: 68000 ASSEMB 64845 01.11

Keywords: ENHANCEMENT

One-line description:
Include support for the ODD psuedo to align data on an odd boundry.

Problem:
The assembler includes support for the EVEN psuedo which alligns data on an even address. It should likewise support the ODD psuedo.

Signed off 04/29/87 in release 501.12

Number: D200059451 Product: 68000 ASSEMB 300 64845S004 01.00

One-line description:
Link_sym file contains bad data in relocatable name record.

Signed off 04/29/87 in release 401.20

Number: D200059477 Product: 68000 ASSEMB 300 64845S004 01.00

One-line description:
Compiler generates duplicate symbols

Problem:
Given a procedure named "AA" and a symbol named "RAA" (or EAA or DAA), the R=label for procedure "AA", "RAA, will collide with the symbol named "RAA" and (at least in the past) no warning will be produced unless the procedure and symbol are GLOBAL, in which case the linker catches the error.

Signed off 04/29/87 in release 401.20

Number: D200059501 Product: 68000 ASSEMB 300 64845S004 01.00

One-line description:
"-v" option does not work with asm inside pmon

Problem:
Note that the status messages do not increment.

Signed off 04/29/87 in release 401.20

Number: D200061713 Product: 68000 ASSEMB 300 64845S004 01.00

One-line description:
External labels cannot be used in the "quick" type instructions.

Problem:
You cannot use an external label as data in the "quick" type instructions. If you have two files:

file: declare

```
"68000"
        GLB     EXTLAB
EXTLAB  EQU     4
```

file: refer

```
"68000"
        EXTERNAL        EXTLAB
LABEL   EQU             7

        MOVEQ.L         #EXTLAB,D1      ;IO error is flagged
        MOVEQ.L         #LABEL,D1       ;WORKS
```

Temporary solution:

Do not use external variables in the "quick" type instructions.
You can possibly get around this by "including" the symbol
(via an include file) rather than declaring it external.

Signed off 04/29/87 in release 401.20

Number: D200066605 Product: 68000 ASSEMB 300 64845S004 01.10

One-line description:
EXT pseudo is not supported as stated in the Assembler reference manual.

Problem:

The Assembler/Linker Reference Manual states that either EXT
or EXTERNAL may be used when declaring an external label.
The 68000 assembler only accepts EXTERNAL.

Temporary solution:
Always use EXTERNAL.

Signed off 04/29/87 in release 401.20

Number: 1650024349 Product: 68000 ASSEMB 500 64845S001 01.50

One-line description:
Assembler reports error if file is specified with full path name.

Problem:
If you try to assemble a file and you specify the file's full
path name the assembler reports:

Cannot recover from errors on line 0.

asm /users/dave/file will cause the error.

Temporary solution:
Work in the same directory as the file is located in. In other
words, assemble the file with its relative path name.

Signed off 04/29/87 in release 101.60

Number: 5000136796 Product: 68000 ASSEMB 500 64845S001 01.40

Keywords: ASSEMBLER

One-line description:
LR error flagged for correct offset using PC+INDEX+OFFSET mode of addr.

Temporary solution:
Temporary solution:

"68000"

```
        ORG     0FFH
        MOVE    TABLE-($+2)[PC,D0],D1
TABLE   DS      1
```

Signed off 04/29/87 in release 101.60

Number: D200061697 Product: 68000 ASSEMB 500 64845S001 01.40

One-line description:
External labels cannot be used in the "quick" type instructions.

Problem:
You cannot use an external label as data in the "quick" type instr-
uctions. If you have two files:

file: declare

"68000"

```
        GLB     EXTLAB
EXTLAB  EQU     4
```

file: refer

"68000"

```
        EXTERNAL        EXTLAB
```

```
LABEL   EQU             7

        MOVEQ.L         #EXTLAB,D1      ;IO error is flagged
        MOVEQ.L         #LABEL,D1       ;WORKS
```

Temporary solution:
Do not use external variables in the "quick" type instructions.
You can possibly get around this by "including" the symbol
(via an include file) rather than declaring it external.

Signed off 04/29/87 in release 101.60

Number: D200062851 Product: 68000 ASSEMB 500 64845S001 01.10

Keywords: LINKER

One-line description:
"Garbage" characters appear in load address statement with linker.

Signed off 04/29/87 in release 101.60

Number: D200066589 Product: 68000 ASSEMB 500 64845S001 01.50

One-line description:
EXT pseudo is not supported as stated in the Assembler reference manual.

Problem:

The Assembler/Linker Reference Manual states that either EXT
or EXTERNAL may be used when declaring an external label.
The 68000 assembler only accepts EXTERNAL.

Temporary solution:
Always use EXTERNAL.

Signed off 04/29/87 in release 101.60

Number: D200061705 Product: 68000 ASSEMB VAX 64845S003 01.50

One-line description:
External labels cannot be used in the "quick" type instructions.

Problem:
You cannot use an external label as data in the "quick" type instr-
uctions. If you have two files:

file: declare

```
"68000"
        GLB     EXTLAB
EXTLAB  EQU     4
```

file: refer

```
"68000"
        EXTERNAL        EXTLAB
LABEL   EQU             7

        MOVEQ.L         #EXTLAB,D1      ;IO error is flagged
        MOVEQ.L         #LABEL,D1       ;WORKS
```

Temporary solution:
Do not use external variables in the "quick" type instructions.
You can possibly get around this by "including" the symbol
(via an include file) rather than declaring it external.

Signed off 04/29/87 in release 301.80

Number: D200066597 Product: 68000 ASSEMB VAX 64845S003 01.70

One-line description:
EXT pseudo is not supported as stated in the Assembler reference manual.

Problem:

The Assembler/Linker Reference Manual states that either EXT
or EXTERNAL may be used when declaring an external label.
The 68000 assembler only accepts EXTERNAL.

Temporary solution:
Always use EXTERNAL.

Signed off 04/29/87 in release 301.80

Number: 5000142331 Product: 68000 C 64819 01.08

One-line description:
Pass three error when an integer is assigned to a float.

Problem:
Compiler generates ERROR 1113 when a condition expression containing real and integer numbers is used.

```
"C"
"processor"

F() {
 float  f1,f2,f3;
 f1=10;
 f2=20;
 f3= (f1<.5) ? 1 : f2;   /* This line is flagged with error 1113. */
}
```

This program will also cause problems on the 6800 and 6809 cross-compilers.

Temporary solution:
In the terenary expression cast the integer '1' to a float or use 1.0.

Signed off 04/29/87 in release 901.10

Number: 5000142448 Product: 68000 C 64819 01.08

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

)
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 901.10

Number: 5000161935 Product: 68000 C 64819 01.09

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:
"C"
"processor"

```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 901.10

Number: D200010124 Product: 68000 C 64819 00.56

Keywords: PASS 1

One-line description:
Unsigned integers treated as signed when subtracted from pointers

Problem:
When an unsigned short or integer is used as an offset to a pointer, the unsigned will be treated as a signed when doing pointer calculations. Offsets large enough to set the sign bit will be interpreted as a negative offset when the offset is subtracted from a pointer. The following code exibits the problem if offset is greater than 32767 dec.

```
unsigned offset;
struct { int a,b,c;
       } *ptr;
unsigned long x;

main ()
```

```
{
  x = ptr - offset;  /* The compiler will generate code negating   */
)                    /* offset for the "-" operation.              */
```

Temporary solution:
Cast the offset in the expression as the next larger integer.
ie. x = ptr - (unsigned long)offset;

Signed off 04/29/87 in release 901.10

Number: D200032029 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:
32 bit value is treated as 64 bit value w/o first extending.

Problem:
In the following C source line the compiler treats the variables as 32 bit values, then in the middle of the compare it treats them as 64 bit values without converting them.

```
"C"
"68000"

main()
{
  float temp;
  temp = ((temp >= 0) ? (temp): (-temp));
}
```

Temporary solution:
Use the alternate "if then" conditional expression.

```
"C"
"68000"

main()
{
 float  temp;

 if (temp <0)
    temp = -temp;
}
```

Signed off 04/29/87 in release 901.10

Number: D200033324 Product: 68000 C 64819 01.07

One-line description:
Defining a constant hex number typecast as a pointer may fail.

Problem:
The following generates incorrect code:

```
"C"
"68000"
typedef char byte;
struct read {
```



```
    byte  int_1;
    byte  int_2;
    byte  int_3;
    };
struct write {
    byte  wrt_1;
    byte  wrt_2;
    byte  wrt_3;
    };
union device {
    struct read  rd;
    struct write wr;
     };
#define PNT ((union device *) 0x80000)
main() {
   PNT->wr.wrt_1 = 1;      /*Generates MOVE.B #00001H,00000H instead of
}                             MOVE.B #00001H,080000H*/
```

This error only occurs when a value is assigned to the first member of either of the structures. It also occurs if PNT is defined as a pointer to a structure, like this:

```
#define PNT ((struct read *) 0x80000)
```

Temporary solution:
There are two possible temporary solutions to this problem. The first is to pad the structures with a dummy variable in the first field.

```
struct read {
   byte  dummy;
   byte  int_1;
   byte  int_2;
   byte  int_3;
   );
```

The second possibility is to use a temporary variable of the appropriate type.

```
main() {
   byte *temp;
   temp = PNT;
   temp->wr.wrt_1 = 1;
}
```

Signed off 04/29/87 in release 901.10

Number: D200033555 Product: 68000 C 64819 01.07

One-line description:
Code generated for return statement inside nested if's is incorrect.

Problem:
In the example below, zero can never be returned due to the code generated (it also could not be generated due to logic of the if statements). In the expanded listing, the code for return zero branches back to the top of the "for" loop rather than exiting.

```
"C"
"68000"

main()
```

```
{
   int i;

        for (i=0; i<i+1; i++)
             if (i>6)
                if (i>4)
                   return(1)
                else return(0)    /* Code generated causes branch to
}                                    top of for loop rather than exiting
                                     function. */
```

Temporary solution:
Enclose the body of the first if statement in braces.

```
"C"
"68000"

main()
{
   int i;
          for (i=0; i< i+1; i++)
              if (i>6) {
                   if (i>4)
                        return(1);
                   else  return(0);
              }
}
```

Signed off 04/29/87 in release 901.10

Number: D200033597 Product: 68000 C 64819 01.07

Keywords: PASS 1

One-line description:
Cannot define a function which returns a pointer to a function.

Problem:
Unable to define a function which returns a pointer to another function.

```
"C"
"68000"

(* x() ) ();  /* Compiler states "Function cannot return a function */

main(){}
```

Signed off 04/29/87 in release 901.10

Number: D200036905 Product: 68000 C 64819 01.07

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.


```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:          break;
                  default:        switch(c){
                                     case 2: break;
                                  }
                                  break;
             }
}
```

Signed off 04/29/87 in release 901.10

Number: D200037358 Product: 68000 C 64819 01.07

Keywords: PASS 1

One-line description:
Wrong value calculated when scientific notation is used.

Problem:
The compiler is calculating incorrect values when scientific notation is used.

```
"C"
"Any Processor"

main(){

     unsigned long  num;

               num = 50.0E+6;        /* Wrong value assigned. */
               num = 50000000;       /* Correct value assigned */
}
```

Temporary solution:

Use the long hand notation.
"C"
"Processor"

```
main(){
      unsigned long num;

       num = 50000000;
}
```

Signed off 04/29/87 in release 901.10

Number: D200052423 Product: 68000 C 64819 01.08

One-line description:
Compiler loads return value in two different locatations.

Problem:
The location for the return values in the below program differ.
This problem is unique to the 68000 C compiler on the 64000.

"C"
"68000"

```
int     a,b,x[5],y;

main() {

  int z;

  return[1);

  /* Problem statement.  Generates code for different return location.*/

  if ((a=((x[b]-2))< 0 ? -1:2)&&(3<4))
     return(0);

}
```

Temporary solution:
Reduce the complexity of the 'if' statement.
"C"
"68000"

```
int a,b,x[5],y;

main()
{
  int z;


 a=(x[b]-2)<0?-1:0 ;

 if(a && (3<4))
   return(1);

}
```


Signed off 04/29/87 in release 901.10

Number: D200053173 Product: 68000 C 64819 01.08

One-line description:
Sign extension done when integer type cast to an unsigned long.

Temporary solution:
You can either typecast the integer to an "unsigned int" or
not typecast it at all.

```
main() {

   int i;
   unsigned long  ul;

   i = 0x8000;
   ul = i;

}
```

Signed off 04/29/87 in release 901.10

Number: D200055921 Product: 68000 C 64819 01.08

Keywords: PASS 3

One-line description:
Incrementing structure member results in incomplete code generation.

Problem:
Dereferencing a pointer within a structure and trying to increment
that pointer causes incomplete code to be generated.

"C"
"68000"

```
main() {

struct {
   int *i,*j;
}*p;

((double*)(p->j))++;          /* Incomplete code is generated. The */
((double*)(p->j)) +=1;        /* The pointer is not incremented.   */
(double*)(p->j) = (double*)(p->j) + 1;

}
```

Temporary solution:
Define a temporary variable, increment the temporary and then
reassign to the original pointer.

"C"
"68000"

```
main() {

   struct {
     int  *i,*j;
   ) *p;

   double *temp;


       temp = (double*)(p-j);
       temp++;
       (double*)(p->j) = temp;

}
```

Signed off 04/29/87 in release 901.10

Number: D200056002 Product: 68000 C 64819 01.08

One-line description:
$INIT_ZEROES$ may affect the addressing mode used for accessing var's.

Problem:
Turning $INIT_ZEROES OFF$ can change the way variables are accessed.

```
"C"
"68000"

static int a;
static int c;

$FAR$
extern b;
f()
{
 a=b;
}
```

If $INIT_ZEROES OFF$ is inserted above the declaration for variable 'a', 'a' will be accessed with a different addressing mode. In the above program 'a' is accessed with the A5 addressing mode, however, if $INIT_ZEROES OFF$ is inserted then 'a' is accessed with the FAR addressing mode.

Note: There appears to be an interaction between the directive INIT ZEROES and the keyword 'static'. If the above program is written as

```
"C"
"68000"

$INIT_ZEROES OFF$
int a;
int c;
$FAR$
```



```
extern int b;
f()
{
 a=b;
}
```

the variable 'a' is accessed properly.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 901.10

Number: D200063420 Product: 68000 C 64819 01.09

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 901.10

Number: D200065144 Product: 68000 C 64819 01.09

One-line description:
Compiler aborts with too many errors in pass 1.

Problem:
The following file will abort in Pass 1 and will report that it cannot recover from errors.

```
"C"
"processor"

extern ident_p();

char *curr_proc_name(pid_ptr)
  unsigned long  pid;

  { char  *proc_pcb;
    long  dummy=0;
    char  status;

    if (ident_p(&dummy1,&proc_pcb,&status))
        return  0L;
  }
```

It is the &dummy1 which causes the abort. If you do not pass the parameter as an address and just misspell the name the correct error message is generated.

Signed off 04/29/87 in release 901.10

Number: D200051458 Product: 68000 C 300 64819S004 01.00

One-line description:
Defining a constant hex number typecast as a pointer may fail.

Problem:
The following generates incorrect code:

```
"C"
"68000"
typedef char byte;
struct read {
    byte  int_1;
    byte  int_2;
    byte  int_3;
    };
struct write {
    byte  wrt_1;
    byte  wrt_2;
    byte  wrt_3;
    };
union device {
    struct read  rd;
    struct write wr;
     };
#define PNT ((union device *) 0x80000)
main() {
   PNT->wr.wrt_1 = 1;     /*Generates MOVE.B #00001H,00000H instead of
}                           MOVE.B #00001H,080000H*/
```

This error only occurs when a value is assigned to the first member of either of the structures. It also occurs if PNT is defined as a pointer to a structure, like this:

```
#define PNT ((struct read *) 0x80000)
```

Temporary solution:
There are two possible temporary solutions to this problem. The first is to pad the structures with a dummy variable in the first field.

```
struct read {
   byte  dummy;
   byte  int_1;
   byte  int_2;
   byte  int_3;
   };
```

The second possibility is to use a temporary variable of the appropriate type.

```
main() {
   byte *temp;
   temp = PNT;
   temp->wr.wrt_1 = 1;
}
```

Signed off 04/29/87 in release 401.20

Number: D200051920 Product: 68000 C 300 64819S004 01.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
       int  c;
          switch(c){
               case1:          break;
               default:        switch(c){
                                  case 2: break;
                               }
                               break;
          }
}
```

Signed off 04/29/87 in release 401.20

Number: D200059493 Product: 68000 C 300 64819S004 01.00

One-line description:
Line # labels emitted for #included files confuse analyzers

Problem:
Line # labels emitted for #include files are ambiguous - there is no indication to analysis tools what the original source file is, so source referencing does not work properly.

In addition, duplicate symbols in any file are debatably errors. As with most symbolic software, EDBUILD will be confused with this behavior.

Signed off 04/29/87 in release 401.20

Number: D200059733 Product: 68000 C 300 64819S004 01.00

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 401.20

Number: D200063453 Product: 68000 C 300 64819S004 01.10

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 401.20

Number: D200064634 Product: 68000 C 300 64819S004 01.10

One-line description:
Byte parameters are pushed onto the stack incorrectly.

Problem:
When passing a byte parameter it is not pushed onto the stack as the manual specifies it will be. The Pascal and C manual specify that a byte parameter will be pushed in the upper byte of the word which is pushed on the stack. The C compiler does a Move.W and pushes the char in the lower byte. The pascal compiler does the push correctly.

```
"C"
"68000"
```



```
char called_func();

calling_func() {

  char  passed_parm;
  passed_parm = 'b';

  called_func(passed_parm);
}

char  called_func(parm)
char  parm;
{
  char local_var;
  local_var = parm;
}
```

Signed off 04/29/87 in release 401.20

Number: D200066035 Product: 68000 C 300 64819S004 01.10

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 401.20

Number: D200066415 Product: 68000 C 300 64819S004 01.10

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 401.20

Number: 1650017491 Product: 68000 C 500 64819S001 01.50

One-line description:
List file contains control characters in a specific case.

Problem:
The following program will cause control characters to be inserted in the listing file. The lines which contain preprocessor substitutions will be effected.

```
"C"
"processor"

#define         varone          0xFE
#define         vartwo          0xFF

extern          int     direct_reg;

write_var()
{
   if (direct_reg == vartwo)        /* Add statement and error goes   */
                                    /* away.                          */

   if (direct_reg != varone)

}
```

As noted in the comment above if the syntax error is removed the problem goes away.

Signed off 04/29/87 in release 101.60

Number: D200033530 Product: 68000 C 500 64819S001 01.10

One-line description:
Defining a constant hex number typecast as a pointer may fail.

Problem:
The following generates incorrect code:

```
"C"
"68000"
typedef char byte;
struct read {
    byte  int_1;
    byte  int_2;
    byte  int_3;
    };
struct write {
    byte  wrt_1;
    byte  wrt_2;
    byte  wrt_3;
    };
union device {
    struct read  rd;
    struct write wr;
```

```
    };
#define PNT ((union device *) 0x80000)
main() {
   PNT->wr.wrt_1 = 1;      /*Generates MOVE.B #00001H,00000H instead of
}                            MOVE.B #00001H,080000H*/
```

This error only occurs when a value is assigned to the first member of either of the structures. It does not occur if PNT is defined as a pointer to a structure, like this:

```
#define PNT ((struct read *) 0x80000)
```

Temporary solution:
There are two possible temporary solutions to this problem. The first is to pad the structures with a dummy variable in the first field.

```
struct read {
   byte  dummy;
   byte  int_1;
   byte  int_2;
   byte  int_3;
   };
```

The second possibility is to use a temporary variable of the appropriate type.

```
main() {
   byte *temp;
   temp = PNT;
   temp->wr.wrt_1 = 1;
}
```

Signed off 04/29/87 in release 101.60

Number: D200040329 Product: 68000 C 500 64819S001 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:

Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                    case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 101.60

Number: D200059717 Product: 68000 C 500 64819S001 01.40

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 101.60

Number: D200063438 Product: 68000 C 500 64819S001 01.50

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 101.60

Number: D200064618 Product: 68000 C 500 64819S001 01.50

One-line description:
Byte parameters are pushed onto the stack incorrectly.

Problem:
When passing a byte parameter it is not pushed onto the stack as the manual specifies it will be. The Pascal and C manual specify that a byte parameter will be pushed in the upper byte of the word which is pushed on the stack. The C compiler does a Move.W and pushes the char in the lower byte. The pascal compiler does the push correctly.

```
"C"
"68000"

char called_func();

calling_func() {

  char  passed_parm;
  passed_parm = 'b';

  called_func(passed_parm);
}

char  called_func(parm)
char  parm;
{
  char local_var;
  local_var = parm;
}
```

Signed off 04/29/87 in release 101.60

Number: D200066019 Product: 68000 C 500 64819S001 01.50

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:

Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 101.60

Number: D200066399 Product: 68000 C 500 64819S001 01.50

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 101.60

Number: 1650019109 Product: 68000 C VAX 64819S003 01.80

One-line description:
Listing file for submitted programs is incomplete.

Problem:
The listing file for the included files is incomplete. The output to standard error is more descriptive than the listing file.

Signed off 04/29/87 in release 301.90

Number: 5000141127 Product: 68000 C VAX 64819S003 01.00

One-line description:
Error message are not consistient.

Problem:
If you have a symbol defined twice (it must be defined in an include file one of the times) different error messages will be flagged depending on the compiler options specified.

```
"C"
"processor"

#define  byte  char        /*Actually defined in an included file. */
#define  byte  int         /* Defined right in source file. */

main() {

int i;

i =5;
}
```

If the above program is compiled using the nocode option you will get a cannot recover from error message and may (depending on the processor) get warning 513. If nocode is not specified the warning 513 is correctly flagged.

Signed off 04/29/87 in release 301.90

Number: D200033548 Product: 68000 C VAX 64819S003 01.20

One-line description:
Defining a constant hex number typecast as a pointer may fail.

Problem:
The following generates incorrect code:

```
"C"
"68000"
typedef char byte;
struct read {
    byte  int_1;
    byte  int_2;
    byte  int_3;
```



```
    };
struct write {
    byte  wrt_1;
    byte  wrt_2;
    byte  wrt_3;
    };
union device {
    struct read  rd;
    struct write wr;
     };
#define PNT ((union device *) 0x80000)
main() {
   PNT->wr.wrt_1 = 1;      /*Generates MOVE.B #00001H,00000H instead of
}                             MOVE.B #00001H,080000H*/
```

This error only occurs when a value is assigned to the first member of either of the structures. It does not occur if PNT is defined as a pointer to a structure, like this:
#define PNT ((struct read *) 0x80000)

Temporary solution:
There are two possible temporary solutions to this problem. The first is to pad the structures with a dummy variable in the first field.

```
struct read {
   byte  dummy;
   byte  int_1;
   byte  int_2;
   byte  int_3;
   };
```

The second possibility is to use a temporary variable of the appropriate type.

```
main() {
   byte *temp;
   temp = PNT;
   temp->wr.wrt_1 = 1;
}
```

Signed off 04/29/87 in release 301.90

Number: D200040337 Product: 68000 C VAX 64819S003 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
```

```
                        }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
            }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 301.90

Number: D200059725 Product: 68000 C VAX 64819S003 01.50

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 301.90

Number: D200063446 Product: 68000 C VAX 64819S003 01.80

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 301.90

Number: D200064626 Product: 68000 C VAX 64819S003 01.80

One-line description:
Byte parameters are pushed onto the stack incorrectly.

Problem:
When passing a byte parameter it is not pushed onto the stack as the manual specifies it will be. The Pascal and C manual specify that a byte parameter will be pushed in the upper byte of the word which is pushed on the stack. The C compiler does a Move.W and pushes the char in the lower byte. The pascal compiler does the push correctly.

```
"C"
"68000"

char called_func();

calling_func() {

  char  passed_parm;
  passed_parm = 'b';

  called_func(passed_parm);
)

char  called_func(parm)
char  parm;
{
  char local_var;
  local_var = parm;
)
```

Signed off 04/29/87 in release 301.90

Number: D200066027 Product: 68000 C VAX 64819S003 01.80

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work");
```

```
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 301.90

Number: D200066407 Product: 68000 C VAX 64819S003 01.80

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 301.90

Number: 1650019406 Product: 6809 C 64822 01.07

One-line description:
Return value of function call is being stored at loc. EMPTYSET.

Problem:
Returning from a function the compiler stores the result to location EMPTY_SET_.

```
"C"
"6809"

$USER_DEFINED$

struct { char value[4]; } typedef lint;

lint funcl()
{
  lint x;
  return(x);
}

main()
{
  lint y;
  y = funcl;
}
```

Return value should be stored at location 'y'. Instead it is stored at EMPTY_SET_.

Signed off 04/29/87 in release 201.08

Number: D200010157 Product: 6809 C 64822 00.06

Keywords: PASS 1

One-line description:
Unsigned integers treated as signed when subtracted from pointers

Problem:
When an unsigned short or integer is used as an offset to a pointer, the unsigned will be treated as a signed when doing pointer calculations. Offsets large enough to set the sign bit will be interpreted as a negative offset when the offset is subtracted from a pointer. The following code exibits the problem if offset is greater than 32767 dec.

```
unsigned offset;
struct { int a,b,c;
       } *ptr;
unsigned long x;

main ()
{
  x = ptr - offset;  /* The compiler will generate code negating   */
}                    /* offset for the "-" operation.              */
```

Temporary solution:

Cast the offset in the expression as the next larger integer.
ie. x = ptr - (unsigned long)offset;

Signed off 04/29/87 in release 201.08

Number: D200011387 Product: 6809 C 64822 00.56

Keywords: PASS 1

One-line description:
Functions invoked via function pointers may JSR the wrong location

Problem:
When the typedef statement is used to define pointers to functions, and this pointer type is used in a cast of a variable array to invoke code stored in that array, program execution may transfer to the wrong location. For example, in the following code the simple call to code_array fails while the call and assignment to p works correctly:

```
typedef int(*PFI)();  /* PFI a pointer to int functions */
int code_array[100];  /* array contains code */
PFI p;                /* p a pointer of type PFI */

pfibug()
{
  (*((PFI) code_array))(); /* fails in JSR to code_array */
  (*(p=(PFI)code_array))(); /* assignment and JSR successful */
}
```

Temporary solution:
Set up a dummy variable and perform an assignment to it when doing this type of operation.

Signed off 04/29/87 in release 201.08

Number: D200040402 Product: 6809 C 64822 01.05

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
```



```
                    evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                    case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 201.08

Number: D200059824 Product: 6809 C 64822 01.06

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 201.08

Number: D200063537 Product: 6809 C 64822 01.07

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 201.08

Number: D200066126 Product: 6809 C 64822 01.07

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:
"C"
"processor"

```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

"C"
"processor"

```
char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 201.08

Number: D200051953 Product: 6809 C 300 64822S004 01.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.
"C"
"6809"

```
main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.
"C"
"6809"

```
main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                      case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 401.20

Number: D200059857 Product: 6809 C 300 64822S004 01.00

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

"C"
"processor"

```
main() {
```

```
int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 401.20

Number: D200063560 Product: 6809 C 300 64822S004 01.10

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 401.20

Number: D200066159 Product: 6809 C 300 64822S004 01.10

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:
"C"
"processor"

```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

"C"
"processor"

```
char  string[]  =  "do it this way";

main()
{
  int i;
```



```
  i = sizeof(string);
}
```

Signed off 04/29/87 in release 401.20

Number: D200066498 Product: 6809 C 300 64822S004 01.10

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 401.20

Number: D200059832 Product: 6809 C 500 64822S001 01.20

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 101.40

Number: D200063545 Product: 6809 C 500 64822S001 01.30

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 101.40

Number: D200066134 Product: 6809 C 500 64822S001 01.30

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
```



```
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 101.40

Number: D200066472 Product: 6809 C 500 64822S001 01.30

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 101.40

Number: D200059840 Product: 6809 C VAX 64822S003 01.20

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 301.60

Number: D200063552 Product: 6809 C VAX 64822S003 01.50

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 301.60

Number: D200066142 Product: 6809 C VAX 64822S003 01.50

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work");
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
```



```
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 301.60

Number: D200066480 Product: 6809 C VAX 64822S003 01.50

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 301.60

Number: 5000098343 Product: 6809 PASCAL 64813 01.08

Keywords: VARIANT RECORDS

One-line description:
Variant records may not work.

Problem:

```
TYPE X = RECORD
     CASE BOOLEAN OF
     TRUE  : (I : INTEGER);
     FALSE : (A : ARRAY[0..1] OF BYTE)
     END;
VAR L : X;
    I2 : INTEGER;
BEGIN
L.I := I2;  {THIS STORE IS MADE VIA THE D-REGISTER}
L.A[0] := 0;  {MEMORY IS CLEARED DIRECTLY}
IF L.I = 5 THEN  { THIS COMPARE IMMEDIATE IS DONE WITH THE D-REGISTER
                   NOT RECOGNIZING THE FACT THAT MEMORY CONTENTS AND
                   THE D-REGISTER ARE DIFFERENT}
```

Temporary solution:
$AMNESIA ON$ around the code.

Signed off 04/29/87 in release 301.11

Number: 5000124065 Product: 6809 PASCAL 64813 01.08

One-line description:
The library routine called DISPOSE does not generate correct code

Problem:
The library routine, DISPOSE, destroys the contents of the U register without restoring it. For example:

```
      TYPE X = RECORD    A :INTEGER;
                         B : ^X;
                         end;
      VAR  P : ^X;
      PROCEDURE TEST (R :X);
      BEGIN
        R := P^;
        NEW(P);
        DISPOSE(P);
        P := R.B;
      END;
      .
```

In this example address of R is store in the U register. WHen the program returns from DISPOSE(P), the U register no longer contains the address of R.

This defect is also reported on the 8086/8 Pascal Compiler (SR#5000124313). In this case, the ES register is being overwritten.

Temporary solution:
No known temporary solution.

Signed off 04/29/87 in release 301.11

Number: D200051573 Product: 6809 PASCAL 300 64813S004 01.00

Keywords: VARIANT RECORDS

One-line description:
Variant records may not work.

Problem:
```
TYPE X = RECORD
     CASE BOOLEAN OF
     TRUE  : (I : INTEGER);
     FALSE : (A : ARRAY[0..1] OF BYTE)
     END;
VAR L : X;
    I2 : INTEGER;
BEGIN
L.I := I2;  {THIS STORE IS MADE VIA THE D-REGISTER}
L.A[0] := 0;  {MEMORY IS CLEARED DIRECTLY}
IF L.I = 5 THEN  { THIS COMPARE IMMEDIATE IS DONE WITH THE D-REGISTER
                   NOT RECOGNIZING THE FACT THAT MEMORY CONTENTS AND
                   THE D-REGISTER ARE DIFFERENT}
```

Temporary solution:
$AMNESIA ON$ around the code.

Signed off 04/29/87 in release 401.20

Number: D200036434 Product: 6809 PASCAL 500 64813S001 01.00

Keywords: VARIANT RECORDS

One-line description:
Variant records may not work.

Problem:
```
TYPE X = RECORD
     CASE BOOLEAN OF
     TRUE  : (I : INTEGER);
     FALSE : (A : ARRAY[0..1] OF BYTE)
     END;
VAR L : X;
    I2 : INTEGER;
BEGIN
L.I := I2;  {THIS STORE IS MADE VIA THE D-REGISTER}
L.A[0] := 0;  {MEMORY IS CLEARED DIRECTLY}
IF L.I = 5 THEN  { THIS COMPARE IMMEDIATE IS DONE WITH THE D-REGISTER
                   NOT RECOGNIZING THE FACT THAT MEMORY CONTENTS AND
                   THE D-REGISTER ARE DIFFERENT}
```

Temporary solution:
$AMNESIA ON$ around the code.

Signed off 04/29/87 in release 101.30

Number: 1650007237 Product: 6809 PASCAL VAX 64813S003 00.00

Keywords: PASS 3

One-line description:
Offset to parameters is incorrect in nested procedure.

Problem:
The 6809 Pascal compiler is pushing parameters incorrectly. Specifically, if several parameters are pushed the 6809 loads the appropiate registers pushes them and begins reloading them. The problem is after pushing the registers the compiler forgets the location of the static link and assumes its at Stack+0.

```
"6809" PREPROCESS

$EXTENSIONS ON$
$RECURSIVE ON$

PROGRAM STACKBUILD;

CONST
   PATTERN_EVENT  = UNSIGNED_8(138);
   NO =  UNSIGNED_8(0);

TYPE
   STRUCTURE  = RECORD
      VAR1    : INTEGER;
      VAR2    : INTEGER;
   END;

   CHAN_PTR   = ^STRUCTURE;
   Q_PTR      = ^STRUCTURE;
   IO_PTR     = ^STRUCTURE;

VAR
  P_PTR  :  Q-PTR;


PROCEDURE   SEND_EVENT(A,B,C,D : UNSIGNED_8; E,F : INTEGER); EXTERNAL;
PROCEDURE   LOG_EVENT(A,B,C,D,E,F : UNSIGNED_8); EXTERNAL;

PROCEDURE ONE_MINUTE_SCAN(P1 : Q_PTR; P2 : IO_PTR);

   VAR
     P3         :    CHAN_PTR;
     OS_TYPE    :    UNSIGNED_8;
     OS_ADDR    :    UNSIGNED_8;
     OS_CHAN    :    UNSIGNED_8;
     T7         :    INTEGER;
     T10        :    INTEGER;


   PROCEDURE   SEND_PATTERN(PATTERN_NO: UNSIGNED_8;
                            X0,X1      : INTEGER;
                            P1         : CHAN_PTR);
```



```
   BEGIN

     SEND_EVENT(PATTERN_EVENT, OS_ADDR, PATTERN_NO, OS_CHAN, X0, X1);
              LDU       07H,S     ;X1
              LDY       05H,S     ;X0
              LDX       00H,S     ;STATIC LINK
              LDB       06H,X     ;OS_CHAN
              LDA       04H,S     ;PATTERN_NO
              PSHS      B,A,Y,U,PC  ;PUSH PARAMETERS
              LDY       00H,S     ; COMPILER THINKS IT IS LOADING
                                  ; STATIC OFFSET.  IT IS ACTUALLY
                                  ; LOADING THE VALUE OF B.
              LDB       0DH,Y     ; GARBAGE
                  .
                  .
                  .
  END; { SEND PATTERN }

BEGIN

   SEND_PATTERN(NO,1,2,P_PTR);
END; { ONE_MINUTE_SCAN }


BEGIN
   ONE_MINUTE_SCAN(P_PTR,P_PTR)

END. { STACKBUILD }
```

Temporary solution:
No known work around at this time.

Signed off 04/29/87 in release 301.40

Number: D200036442 Product: 6809 PASCAL VAX 64813S003 01.00

Keywords: VARIANT RECORDS

One-line description:
Variant records may not work.

Problem:

```
TYPE X = RECORD
     CASE BOOLEAN OF
     TRUE  : (I : INTEGER);
     FALSE : (A : ARRAY[0..1] OF BYTE)
     END;
VAR L : X;
    I2 : INTEGER;
BEGIN
L.I := I2;  {THIS STORE IS MADE VIA THE D-REGISTER}
L.A[0] := 0;  {MEMORY IS CLEARED DIRECTLY}
IF L.I = 5 THEN  { THIS COMPARE IMMEDIATE IS DONE WITH THE D-REGISTER
                   NOT RECOGNIZING THE FACT THAT MEMORY CONTENTS AND
                   THE D-REGISTER ARE DIFFERENT}
```

Temporary solution:
$AMNESIA ON$ around the code.

Signed off 04/29/87 in release 301.40

Number: D200067561 Product: 80286B ASSEMB 64859 01.00

Keywords: LINKER

One-line description:
Error flag not set when file required by link is missing

Problem:
System error flag not set (command doesn't stop) when a file required by a link is missing.

Signed off 04/29/87 in release 901.02

Number: D200067579 Product: 80286B ASSEMB 64859 01.00

Keywords: ENHANCEMENT

One-line description:
Seperate linker outputs by adding several blank lines at the start

Problem:
The 64100 80286B linker output runs together when displayed. The beginning of each linker output should have several blank lines added so that the beginning of the new linker output can be determined. Another alternativce would be to clear the screen at the beginning of each linker output.

Signed off 04/29/87 in release 901.02

Number: D200067595 Product: 80286B ASSEMB 64859 01.00

Keywords: ENHANCEMENT

One-line description:
Change the linker to only accept 80286B link_sym files

Signed off 04/29/87 in release 901.02

Number: D200067603 Product: 80286B ASSEMB 64859 01.00

Keywords: ENHANCEMENT

One-line description:
File with unsupported processor name should be specified in error msg

Signed off 04/29/87 in release 901.02

Number: D200067611 Product: 80286B ASSEMB 64859 01.00

Keywords: ENHANCEMENT

One-line description:
Warning message should be generated when aliasing an alias

Problem:
Enhancement request to add a warning message when aliasing an alias.

Signed off 04/29/87 in release 901.02

Number: D200055426 Product: 80286B ASSEMB 300 64859S004 01.00

Keywords: CODE GENERATOR

One-line description:
FSTSW/FNSTSW function incorrectly with two-byte memory operand

Problem:
FSTSW/FNSTSW (Store 80287 Status Word) instruction incorrectly results in Invalid Operand Error when used with two-byte memory operand. This instruction should accept a two-byte memory operand.

Temporary solution:
The FSTSW AX or FNSTSW AX versions of the FSTSW/FNSTSW instructions can be used followed immediately by the MOV mem,AX instruction.

Signed off 04/29/87 in release 401.10

Number: D200055459 Product: 80286B ASSEMB 300 64859S004 01.00

Keywords: CODE GENERATOR

One-line description:
FSTENV instruction generates object code without required wait instr

Problem:
The object code for the FSTENV instruction is missing the required wait instruction. The code generated is D936001A, it should be 9BD936001A.

Temporary solution:
Precede all FSTENV instructions with the WAIT instruction.

Signed off 04/29/87 in release 401.10

Number: D200055483 Product: 80286B ASSEMB 300 64859S004 01.00

Keywords: CODE GENERATOR

One-line description:
Obj. code generated for arithmetic instr. are incorrect.

Problem:
The object code produced for the arithmetic instructions FADD,FDIV, FDIVP, FDIVR, FDIVRP, FMUL, FSUB, FSUBP, FSUBR, FSUBRP is not correct. These problems occur only with the code generated for the 80287 coprocessor. The 8087 processor was changed in Feb. 1984. The opcodes generated are for 8087 processors manufactured prior to Feb. 1984.

instruction		opcode - valid prior to FEB. 1984
FADD		DCC1
FDIV		DCF1
FDIV	ST[3],ST	DCF3
FDIVP	ST[4],ST	DEF4
FDIVR		DCF9
FDIVR	ST[4],ST	DCFC
FDIVRP	ST[1],ST	DEF9

```
          FMUL                  DCC9
          FSUB                  DCE1
          FSUB    ST[1],ST      DCE2
          FSUBP   ST[2],ST      DEE2
          FSUBR                 DCE9
          FSUBR   ST[1],ST      DCE9
          FSUBRP  ST[1],ST      DEE9
```

The pseudo instruction NEW_8087 will cause the correct opcodes to be generated. This assembler should default to the new opcodes without the pseudo instruction.

Temporary solution:
The "NEW_8087" pseudo instruction should be included in any program using the 80287 arithmetic instructions. This pseudo instruction should precede all 80287 instructions.

ex. "80286B"

```
                        NEW_8087

                        PROG

     DEC1               FADD
     DEF9               FDIV
     DCFB               FDIV    ST[3],ST
     DEFC               FDIVP   ST[4],ST
     DEF1               FDIVR
     DCF4               FDIVR   ST[4],ST
     DEF1               FDIVRP  ST[1],ST
     DEC9               FMUL
     DEE9               FSUB
     DCE9               FSUB    ST[1],ST
     DEEA               FSUBP   ST[2],ST
     DEE1               FSUBR
     DCE1               FSUBR   ST[1],ST
     DEE1               FSUBRP  ST[1],ST
```

Signed off 04/29/87 in release 401.10

Number: D200055400 Product: 80286B ASSEMB 500 64859S001 01.00

Keywords: CODE GENERATOR

One-line description:
FSTSW/FNSTSW function incorrectly with two-byte memory operand

Problem:
FSTSW/FNSTSW (Store 80287 Status Word) instruction incorrectly results in Invalid Operand Error when used with two-byte memory operand. This instruction should accept a two-byte memory operand.

Temporary solution:
The FSTSW AX or FNSTSW AX versions of the FSTSW/FNSTSW instructions can be used followed immediately by the MOV mem,AX instruction.

Signed off 04/29/87 in release 101.10

Number: D200055434 Product: 80286B ASSEMB 500 64859S001 01.00

Keywords: CODE GENERATOR

One-line description:
FSTENV instruction generates object code without required wait instr

Problem:
The object code for the FSTENV instruction is missing the required wait instruction. The code generated is D936001A, it should be 9BD936001A.

Temporary solution:
Precede all FSTENV instructions with the WAIT instruction.

Signed off 04/29/87 in release 101.10

Number: D200055467 Product: 80286B ASSEMB 500 64859S001 01.00

Keywords: CODE GENERATOR

One-line description:
Obj. code generated for arithmetic instr. are incorrect.

Problem:
The object code produced for the arithmetic instructions FADD,FDIV, FDIVP, FDIVR, FDIVRP, FMUL, FSUB, FSUBP, FSUBR, FSUBRP is not correct. These problems occur only with the code generated for the 80287 coprocessor. The 8087 processor was changed in Feb. 1984. The opcodes generated are for 8087 processors manufactured prior to Feb. 1984.

```
          instruction           opcode - valid prior to FEB. 1984

          FADD                  DCC1
          FDIV                  DCF1
          FDIV    ST[3],ST      DCF3
          FDIVP   ST[4],ST      DEF4
          FDIVR                 DCF9
          FDIVR   ST[4],ST      DCFC
          FDIVRP  ST[1],ST      DEF9
```

```
          FMUL                DCC9
          FSUB                DCE1
          FSUB    ST[1],ST    DCE2
          FSUBP   ST[2],ST    DEE2
          FSUBR               DCE9
          FSUBR   ST[1],ST    DCE9
          FSUBRP  ST[1],ST    DEE9
```

The pseudo instruction NEW_8087 will cause the correct opcodes to be generated. This assembler should default to the new opcodes without the pseudo instruction.

Temporary solution:
The "NEW_8087" pseudo instruction should be included in any program using the 80287 arithmetic instructions. This pseudo instruction should precede all 80287 instructions.

ex. "80286B"

```
                    NEW_8087

                    PROG

      DEC1          FADD
      DEF9          FDIV
      DCFB          FDIV    ST[3],ST
      DEFC          FDIVP   ST[4],ST
      DEF1          FDIVR
      DCF4          FDIVR   ST[4],ST
      DEF1          FDIVRP  ST[1],ST
      DEC9          FMUL
      DEE9          FSUB
      DCE9          FSUB    ST[1],ST
      DEEA          FSUBP   ST[2],ST
      DEE1          FSUBR
      DCE1          FSUBR   ST[1],ST
      DEE1          FSUBRP  ST[1],ST
```

Signed off 04/29/87 in release 101.10

Number: D200055418 Product: 80286B ASSEMB VAX 64859S003 01.00

Keywords: CODE GENERATOR

One-line description:
FSTSW/FNSTSW function incorrectly with two-byte memory operand

Problem:
FSTSW/FNSTSW (Store 80287 Status Word) instruction incorrectly results in Invalid Operand Error when used with two-byte memory operand. This instruction should accept a two-byte memory operand.

Temporary solution:
The FSTSW AX or FNSTSW AX versions of the FSTSW/FNSTSW instructions can be used followed immediately by the MOV mem,AX instruction.

Signed off 04/29/87 in release 301.10

Number: D200055442 Product: 80286B ASSEMB VAX 64859S003 01.00

Keywords: CODE GENERATOR

One-line description:
FSTENV instruction generates object code without required wait instr

Problem:
The object code for the FSTENV instruction is missing the required wait instruction. The code generated is D936001A, it should be 9BD936001A.

Temporary solution:
Precede all FSTENV instructions with the WAIT instruction.

Signed off 04/29/87 in release 301.10

Number: D200055475 Product: 80286B ASSEMB VAX 64859S003 01.00

Keywords: CODE GENERATOR

One-line description:
Obj. code generated for arithmetic instr. are incorrect.

Problem:
The object code produced for the arithmetic instructions FADD,FDIV, FDIVP, FDIVR, FDIVRP, FMUL, FSUB, FSUBP, FSUBR, FSUBRP is not correct. These problems occur only with the code generated for the 80287 coprocessor. The 8087 processor was changed in Feb. 1984. The opcodes generated are for 8087 processors manufactured prior to Feb. 1984.

```
          instruction          opcode - valid prior to FEB. 1984

          FADD                 DCC1
          FDIV                 DCF1
          FDIV    ST[3],ST     DCF3
          FDIVP   ST[4],ST     DEF4
          FDIVR                DCF9
          FDIVR   ST[4],ST     DCFC
          FDIVRP  ST[1],ST     DEF9
```

```
          FMUL                  DCC9
          FSUB                  DCE1
          FSUB    ST[1],ST      DCE2
          FSUBP   ST[2],ST      DEE2
          FSUBR                 DCE9
          FSUBR   ST[1],ST      DCE9
          FSUBRP  ST[1],ST      DEE9
```

The pseudo instruction NEW_8087 will cause the correct opcodes to be generated. This assembler should default to the new opcodes without the pseudo instruction.

Temporary solution:
The "NEW_8087" pseudo instruction should be included in any program using the 80287 arithmetic instructions. This pseudo instruction should precede all 80287 instructions.

ex. "80286B"

```
                        NEW_8087

                        PROG

          DEC1          FADD
          DEF9          FDIV
          DCFB          FDIV    ST[3],ST
          DEFC          FDIVP   ST[4],ST
          DEF1          FDIVR
          DCF4          FDIVR   ST[4],ST
          DEF1          FDIVRP  ST[1],ST
          DEC9          FMUL
          DEE9          FSUB
          DCE9          FSUB    ST[1],ST
          DEEA          FSUBP   ST[2],ST
          DEE1          FSUBR
          DCE1          FSUBR   ST[1],ST
          DEE1          FSUBRP  ST[1],ST
```

Signed off 04/29/87 in release 301.10

Number: D200067546 Product: 80286B ASSEMB VAX 64859S003 01.00

One-line description:
Build files generated on the VAX will not work with the 286 linker

Problem:
The 286 builder does not function correctly when using build files that were created on the VAX. These files are record format which contains EOF errors; therefore, all build files generated on the VAX will fail. The build files generated on the 64100 are uploaded as streamlf format files and ,therefore, do not contain this problem.

Temporary solution:
Temporary solution: Generate build files on the 64100 and upload to the VAX.

Signed off 04/29/87 in release 301.10

Number: D200016295 Product: 8085 8 PASCAL 64825 01.01

Keywords: CODE GENERATOR

One-line description:
Compiler generates incorrect code (assignment to record variable).

Temporary solution:
Enable the compiler option AMNESIA before the assignment statement.

Signed off 04/29/87 in release 501.04

Number: D200020081 Product: 8085 B PASCAL 64825 01.01

One-line description:
Compiler does not generate cross reference table.

Problem:
When compiling a program using the option 'xref', no cross reference table will be generated if the compilation completes without errors.

Temporary solution:
To generate a cross reference table simply edit the source file and introduce an error (syntax error will do). The error will cause the compiler to generate the cross reference table. Once the table has been generated simply edit the source file and remove the error.

Signed off 04/29/87 in release 501.04

Number: D200029793 Product: 8085 8 PASCAL 64825 01.01

Keywords: POINTERS

One-line description:
Variables of type pointer may not be incremented correctly.

Problem:
```
"PROCESSOR"
TYPE
PTR = ^BYTE;
TX  = PTR;

VAR
RXOUT: TX;
TEMP1,TEMP2 : BYTE;

8EGIN
TEMP1 := RXOUT^;
     LD     HL,[RXOUT]
     LD     A,[HL]
     LD     [TEMP1], A   ;HERE, TEMP1 IS CORRECTLY LOADED WITH THE BYTE
                         ;THAT RXOUT IS POINTING TO

RXOUT := TX(SIGNED_16(RXOUT)+1); {INCREMENT RXOUT}
     LD     HL,[RXOUT]
```



```
     INC    HL
     LD     [RXOUT],HL ;RXOUT IS CORRECTLY INCREMENTED

TEMP2 := RXOUT^;  {TEMP2 SHOULD GET THE NEXT BYTE}
     LD     [TEMP2],A  ;SINCE A WAS NOT DISTURBED, THE COMPILER DOES
                       ;NOT REALIZE THAT THE POINTER WAS UPDATED.
```

Temporary solution:
Set $AMNESIA ON$ around the pointer referencing code.

Signed off 04/29/87 in release 501.04

Number: D200036863 Product: 8085 8 PASCAL 64825 01.01

Keywords: IF

One-line description:
IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}

Problem:
```
VAR 81, 82 : 8YTE;

8EGIN
IF 81 (>|<|=|<=|>=) B2 THEN
81 := B1 - 1;   {THE REGISTER CONTAINING B1 IS DECREMENTED, THEN
                 OVERWRITTEN BEFORE IT IS SAVED IN MEMORY}
```

Temporary solution:
$AMNESIA +$

Signed off 04/29/87 in release 501.04

Number: D200040121 Product: 8085 B PASCAL 64825 01.01

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for adding one char to another.

Problem:
```
VAR
SRC, DEST : CHAR;

8EGIN
DEST := DEST + SRC;  {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 04/29/87 in release 501.04

Number: D200041137 Product: 8085 B PASCAL 64825 01.01

Keywords: PASS 2

One-line description:
REBOOT DURING PASS 2

Problem:
The 64000 will reboot during pass 2 when compiling files where
1) The 105th external variable is an array, and
2) An element of the 105th external variable is accessed in the 19th procedure or function in the file (external and locally defined procedures count in this total).

Temporary solution:
Change the order of the external variable declarations, or change the order of the procedure declarations.

Signed off 04/29/87 in release 501.04

Number: D200064329 Product: 8085 B PASCAL 64825 01.03

One-line description:
Error #1009 using byte-sized ORG'ed variables in FOR loops

Problem:
Error #1009 is generated when byte sized ORG'ed variables are used in FOR loops. The following code illustrates the problem.

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
PROCEDURE ERR;
VAR
$ORG 5000$
   B1,B2,X1: BYTE;

BEGIN

   FOR X1 := B1 to B2 DO;     (*Pass 2 Error 1009 - No free registers*)
END;
.
```

Temporary solution:
The error does not occur if the FOR loop variable is word sized instead of byte sized. It will also go away if the ORG statement is removed.

Signed off 04/29/87 in release 501.04

Number: D20006442B Product: B0B5 B PASCAL 64825 01.03

One-line description:
32-bit unsigned divide and modulus may fail

Problem:
The result of an unsigned 32-bit division or modulus operation may be incorrect if the dividend and the destination are the same

location. The problem is in the library routine Zdworddiv. The following code demonstrates the problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2 : UNSIGNED_32;
BEGIN
  B1 := UNSIGNED_32(0E00000000);
  B2 := UNSIGNED_32(0900000000);
  B1 := B1/B2;
END.
```

Signed off 04/29/87 in release 501.04

Number: D200064493 Product: 8085 B PASCAL 64825 01.03

One-line description:
Library routine REAL_ROUND may fail.

Problem:
The library routine REAL_ROUND may fail, causing floating point numbers to be incorrectly rounded to integers.

Signed off 04/29/87 in release 501.04

Number: D200064568 Product: 8085 B PASCAL 64825 01.03

One-line description:
DEBUG byte division and modulus may incorrectly report division by zero

Problem:
The DEBUG library routines for performing signed and unsigned byte division and modulus operations may fail and incorrectly report an attempted division by zero.

The following code fails in this manner:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2,B3 : BYTE;
$ORG 5000H$
  BA : ARRAY[1..15] OF BYTE;

BEGIN
  B1 := 1;
  B2 := 1;
  B3 := 0;
  BA[B3] := B1 DIV B2;   (*DIV fails - reports division by zero*)
END.
```

Signed off 04/29/87 in release 501.04

Number: D200064956 Product: 8085 B PASCAL 64825 01.03

One-line description:
Set comparisons with the empty set may fail

Problem:
Set comparisons with the empty set may fail. The following code is an example of this problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
TYPE
  CH = 0..127;
  SET1 = SET OF CH;
VAR
  S1 : SET1;
PROCEDURE ERROR; EXTERNAL;
BEGIN
  S1 := [];
  IF S1 <> [] THEN          (*In CONST_prog, not enough bytes are
    ERROR;                    defined for the set*)
END.
```

Signed off 04/29/87 in release 501.04

Number: D200065326 Product: 8085 B PASCAL 64825 01.03

One-line description:
Assignment of constant string of length 1 to string variable may fail.

Problem:
Assignment of a constant string of length 1 to a string variable that is itself a multidimensional array element may fail.

First, the address of the destination string is calculated in HL. Then the value of the string length resulting from the assignment, i.e. one (1), is loaded into the position reserved for the length of the string via a store indirect through HL. Up to this point all is as it should be; however, the value of the single character that comprises the string is then also stored HL indirect, overwriting the length and failing to correctly load the string value. The HL register should be incremented before the second store.

The following is an example:

```
"processor name"
PROGRAM TEST;
TYPE
  STRING_15 = PACKED ARRAY[0..15] OF CHAR;
VAR
  TWO_D_ARR : ARRAY[1..3,1..3] OF STRING_15;
BEGIN
  TWO_D_ARR[2,1] := " ";
        LD   HL,0030H
        PUSH HL
        LD   HL,00002H
```



```
        PUSH HL
        LD   HL,00010H
        PUSH HL
        LD   HL,00001H
        PUSH HL
        LD   BC,DTEST-00040H
        LD   A,002H
        CALL Zarrayref
        LD   A,001H
        LD   [HL],A       (*or LD   M,A  *)
        LD   A,020H
        LD   [HL],A        (*This is the error - should INC HL first*)
END.
```

Signed off 04/29/87 in release 501.04

Number: D200050203 Product: 8085 B PASCAL 300 64825S004 01.00

Keywords: CODE GENERATOR

One-line description:
Compiler generates incorrect code (assignment to record variable).

Problem:
The following program causes incorrect code to be generated
for the second assignment statement.

```
PROGRAM SCAN;
$EXTENSIONS ON$

TYPE
  INTEGER = SIGNED_16;
  HEAD_STRUC = RECORD
               NUM: INTEGER;
               TFPSNO: INTEGER;
               TMPSNO: INTEGER;
  END;

VAR
  HEAD: ARRAY[1..6] OF HEAD_STRUC;

BEGIN
  HEAD[1].NUM:= 1;
  HEAD[1].TFPSNO:= 3;
END.
```

Temporary solution:
Enable the compiler option AMNESIA before the assignment statement.

Signed off 04/29/87 in release 401.20

Number: D200051094 Product: 8085 B PASCAL 300 64825S004 01.00

Keywords: POINTERS

One-line description:
Variables of type pointer may not be incremented correctly.

Problem:

```
"PROCESSOR"
TYPE
PTR = ^BYTE;
TX  = PTR;

VAR
RXOUT: TX;
TEMP1,TEMP2 : BYTE;

BEGIN
TEMP1 := RXOUT^;
     LD     HL,[RXOUT]
     LD     A,[HL]
     LD     [TEMP1], A   ;HERE, TEMP1 IS CORRECTLY LOADED WITH THE BYTE
```



```
                         ;THAT RXOUT IS POINTING TO

RXOUT := TX(SIGNED_16(RXOUT)+1); {INCREMENT RXOUT}
     LD     HL,[RXOUT]
     INC    HL
     LD     [RXOUT],HL ;RXOUT IS CORRECTLY INCREMENTED

TEMP2 := RXOUT^;  {TEMP2 SHOULD GET THE NEXT BYTE}
     LD     [TEMP2],A  ;SINCE A WAS NOT DISTURBED, THE COMPILER DOES
                       ;NOT REALIZE THAT THE POINTER WAS UPDATED.
```

Temporary solution:
Set $AMNESIA ON$ around the pointer referencing code.

Signed off 04/29/87 in release 401.20

Number: D200051607 Product: 8085 B PASCAL 300 64825S004 01.00

Keywords: IF

One-line description:
IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}

Problem:

```
VAR B1, B2 : BYTE;

BEGIN
IF B1 (>|<|=|<=|>=) B2 THEN
B1 := B1 - 1;   {THE REGISTER CONTAINING B1 IS DECREMENTED, THEN
                 OVERWRITTEN BEFORE IT IS SAVED IN MEMORY}
```

Temporary solution:

```
$AMNESIA +$
```

Signed off 04/29/87 in release 401.20

Number: D200051862 Product: 8085 B PASCAL 300 64825S004 01.00

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for adding one char to another.

Problem:

```
VAR
SRC, DEST : CHAR;

BEGIN
DEST := DEST + SRC;  {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 04/29/87 in release 401.20

Number: D200064352 Product: 8085 B PASCAL 300 64825S004 01.10

One-line description:
Error #1009 using byte-sized ORG'ed variables in FOR loops

Problem:
Error #1009 is generated when byte sized ORG'ed variables are used in FOR loops. The following code illustrates the problem.

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
PROCEDURE ERR;
VAR
$ORG 5000$
   B1,B2,X1: BYTE;

BEGIN

   FOR X1 := B1 to B2 DO;     (*Pass 2 Error 1009 - No free registers*)
END;
.
```

Temporary solution:
The error does not occur if the FOR loop variable is word sized instead of byte sized. It will also go away if the ORG statement is removed.

Signed off 04/29/87 in release 401.20

Number: D200064451 Product: 8085 B PASCAL 300 64825S004 01.10

One-line description:
32-bit unsigned divide and modulus may fail

Problem:
The result of an unsigned 32-bit division or modulus operation may be incorrect if the dividend and the destination are the same location. The problem is in the library routine Zdworddiv. The following code demonstrates the problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2 : UNSIGNED_32;
BEGIN
  B1 := UNSIGNED_32(0E00000000);
  B2 := UNSIGNED_32(0900000000);
  B1 := B1/B2;
END.
```

Signed off 04/29/87 in release 401.20

Number: D200064527 Product: 8085 B PASCAL 300 64825S004 01.10

One-line description:
Library routine REAL_ROUND may fail.

Problem:
The library routine REAL_ROUND may fail, causing floating point numbers to be incorrectly rounded to integers.

Signed off 04/29/87 in release 401.20

Number: D200064592 Product: 8085 B PASCAL 300 64825S004 01.10

One-line description:
DEBUG byte division and modulus may incorrectly report division by zero

Problem:
The DEBUG library routines for performing signed and unsigned byte division and modulus operations may fail and incorrectly report an attempted division by zero.

The following code fails in this manner:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2,B3 : BYTE;
$ORG 5000H$
  BA : ARRAY[1..15] OF BYTE;

BEGIN
  B1 := 1;
  B2 := 1;
  B3 := 0;
  BA[B3] := B1 DIV B2;   (*DIV fails - reports division by zero*)
END.
```

Signed off 04/29/87 in release 401.20

Number: D200064980 Product: 8085 B PASCAL 300 64825S004 01.10

One-line description:
Set comparisons with the empty set may fail

Problem:
Set comparisons with the empty set may fail. The following code is an example of this problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
TYPE
  CH = 0..127;
  SET1 = SET OF CH;
VAR
  S1 : SET1;
```

```
PROCEDURE ERROR; EXTERNAL;
BEGIN
  S1 := [];
  IF S1 <> [] THEN          (*In CONST_prog, not enough bytes are
    ERROR;                    defined for the set*)
END.
```

Signed off 04/29/87 in release 401.20

Number: D200065359 Product: 8085 B PASCAL 300 64825S004 01.10

One-line description:
Assignment of constant string of length 1 to string variable may fail.

Problem:
Assignment of a constant string of length 1 to a string variable that is itself a multidimensional array element may fail.

First, the address of the destination string is calculated in HL. Then the value of the string length resulting from the assignment, i.e. one (1), is loaded into the position reserved for the length of the string via a store indirect through HL. Up to this point all is as it should be; however, the value of the single character that comprises the string is then also stored HL indirect, overwriting the length and failing to correctly load the string value. The HL register should be incremented before the second store.

The following is an example:

```
"processor name"
PROGRAM TEST;
TYPE
  STRING_15 = PACKED ARRAY[0..15] OF CHAR;
VAR
  TWO_D_ARR : ARRAY[1..3,1..3] OF STRING_15;
BEGIN
  TWO_D_ARR[2,1] := " ";
        LD   HL,0030H
        PUSH HL
        LD   HL,00002H
        PUSH HL
        LD   HL,00010H
        PUSH HL
        LD   HL,00001H
        PUSH HL
        LD   BC,DTEST-00040H
        LD   A,002H
        CALL Zarrayref
        LD   A,001H
        LD   [HL],A       (*or LD   M,A  *)
        LD   A,020H
        LD   [HL],A        (*This is the error - should INC HL first*)
END.
```

Signed off 04/29/87 in release 401.20

Number: D200016311 Product: 8085 B PASCAL 500 64825S001 01.10

Keywords: CODE GENERATOR

One-line description:
Compiler generates incorrect code (assignment to record variable).

Temporary solution:
Enable the compiler option AMNESIA before the assignment statement.

Signed off 04/29/87 in release 101.50

Number: D200029801 Product: 8085 B PASCAL 500 64825S001 01.10

Keywords: POINTERS

One-line description:
Variables of type pointer may not be incremented correctly.

Problem:

```
"PROCESSOR"
TYPE
PTR = ^BYTE;
TX  = PTR;

VAR
RXOUT: TX;
TEMP1,TEMP2 : BYTE;

BEGIN
TEMP1 := RXOUT^;
     LD     HL,[RXOUT]
     LD     A,[HL]
     LD     [TEMP1], A   ;HERE, TEMP1 IS CORRECTLY LOADED WITH THE BYTE
                         ;THAT RXOUT IS POINTING TO

RXOUT := TX(SIGNED_16(RXOUT)+1); {INCREMENT RXOUT}
     LD     HL,[RXOUT]
     INC    HL
     LD     [RXOUT],HL ;RXOUT IS CORRECTLY INCREMENTED

TEMP2 := RXOUT^;  {TEMP2 SHOULD GET THE NEXT BYTE}
     LD     [TEMP2],A  ;SINCE A WAS NOT DISTURBED, THE COMPILER DOES
                       ;NOT REALIZE THAT THE POINTER WAS UPDATED.
```

Temporary solution:
Set $AMNESIA ON$ around the pointer referencing code.

Signed off 04/29/87 in release 101.50

Number: D200036848 Product: 8085 B PASCAL 500 64825S001 01.20

Keywords: IF

One-line description:
IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}

Problem:
VAR B1, B2 : BYTE;

```
BEGIN
IF B1 (>|<|=|<=|>=) B2 THEN
B1 := B1 - 1;   {THE REGISTER CONTAINING B1 IS DECREMENTED, THEN
                 OVERWRITTEN BEFORE IT IS SAVED IN MEMORY}
```

Temporary solution:
$AMNESIA +$

Signed off 04/29/87 in release 101.50

Number: D200040139 Product: 8085 B PASCAL 500 64825S001 01.20

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for adding one char to another.

Problem:

```
VAR
SRC, DEST : CHAR;

BEGIN
DEST := DEST + SRC;  {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 04/29/87 in release 101.50

Number: D200064337 Product: 8085 B PASCAL 500 64825S001 01.40

One-line description:
Error #1009 using byte-sized ORG'ed variables in FOR loops

Problem:
Error #1009 is generated when byte sized ORG'ed variables are used in FOR loops. The following code illustrates the problem.

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
PROCEDURE ERR;
VAR
$ORG 5000$
   B1,B2,X1: BYTE;

BEGIN

   FOR X1 := B1 to B2 DO;     (*Pass 2 Error 1009 - No free registers*)
END;
.
```

Temporary solution:

The error does not occur if the FOR loop variable is word sized instead of byte sized. It will also go away if the ORG statement is removed.

Signed off 04/29/87 in release 101.50

Number: D200064436 Product: 8085 B PASCAL 500 64825S001 01.40

One-line description:
32-bit unsigned divide and modulus may fail

Problem:
The result of an unsigned 32-bit division or modulus operation may be incorrect if the dividend and the destination are the same location. The problem is in the library routine Zdworddiv. The following code demonstrates the problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2 : UNSIGNED_32;
BEGIN
  B1 := UNSIGNED_32(0E00000000);
  B2 := UNSIGNED_32(0900000000);
  B1 := B1/B2;
END.
```

Signed off 04/29/87 in release 101.50

Number: D200064501 Product: 8085 B PASCAL 500 64825S001 01.40

One-line description:
Library routine REAL_ROUND may fail.

Problem:
The library routine REAL_ROUND may fail, causing floating point numbers to be incorrectly rounded to integers.

Signed off 04/29/87 in release 101.50

Number: D200064576 Product: 8085 B PASCAL 500 64825S001 01.40

One-line description:
DEBUG byte division and modulus may incorrectly report division by zero

Problem:
The DEBUG library routines for performing signed and unsigned byte division and modulus operations may fail and incorrectly report an attempted division by zero.

The following code fails in this manner:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2,B3 : BYTE;
```

```
$ORG 5000H$
  BA : ARRAY[1..15] OF BYTE;

BEGIN
  B1 := 1;
  B2 := 1;
  B3 := 0;
  BA[B3] := B1 DIV B2;   (*DIV fails - reports division by zero*)
END.
```

Signed off 04/29/87 in release 101.50

Number: D200064964 Product: 8085 B PASCAL 500 64825S001 01.40

One-line description:
Set comparisons with the empty set may fail

Problem:
Set comparisons with the empty set may fail. The following code is an example of this problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
TYPE
  CH = 0..127;
  SET1 = SET OF CH;
VAR
  S1 : SET1;
PROCEDURE ERROR; EXTERNAL;
BEGIN
  S1 := [];
  IF S1 <> [] THEN          (*In CONST_prog, not enough bytes are
    ERROR;                    defined for the set*)
END.
```

Signed off 04/29/87 in release 101.50

Number: D200065334 Product: 8085 B PASCAL 500 64825S001 01.40

One-line description:
Assignment of constant string of length 1 to string variable may fail.

Problem:
Assignment of a constant string of length 1 to a string variable that is itself a multidimensional array element may fail.

First, the address of the destination string is calculated in HL. Then the value of the string length resulting from the assignment, i.e. one (1), is loaded into the position reserved for the length of the string via a store indirect through HL. Up to this point all is as it should be; however, the value of the single character that comprises the string is then also stored HL indirect, overwriting the length and failing to correctly load the string value. The HL register should be incremented before the second store.

The following is an example:


```
"processor name"
PROGRAM TEST;
TYPE
  STRING_15 = PACKED ARRAY[0..15] OF CHAR;
VAR
  TWO_D_ARR : ARRAY[1..3,1..3] OF STRING_15;
BEGIN
  TWO_D_ARR[2,1] := " ";
        LD   HL,0030H
        PUSH HL
        LD   HL,00002H
        PUSH HL
        LD   HL,00010H
        PUSH HL
        LD   HL,00001H
        PUSH HL
        LD   BC,DTEST-00040H
        LD   A,002H
        CALL Zarrayref
        LD   A,001H
        LD   [HL],A      (*or LD   M,A  *)
        LD   A,020H
        LD   [HL],A       (*This is the error - should INC HL first*)
END.
```

Signed off 04/29/87 in release 101.50

Number: D200016303 Product: 8085 B PASCAL VAX 64825S003 01.10

Keywords: CODE GENERATOR

One-line description:
Compiler generates incorrect code (assignment to record variable).

Temporary solution:
Enable the compiler option AMNESIA before the assignment statement.

Signed off 04/29/87 in release 301.70

Number: D200029819 Product: 8085 B PASCAL VAX 64825S003 01.20

Keywords: POINTERS

One-line description:
Variables of type pointer may not be incremented correctly.

Problem:

```
"PROCESSOR"
TYPE
PTR = ^BYTE;
TX  = PTR;

VAR
RXOUT: TX;
TEMP1,TEMP2 : BYTE;

BEGIN
TEMP1 := RXOUT^;
     LD     HL,[RXOUT]
     LD     A,[HL]
     LD     [TEMP1], A   ;HERE, TEMP1 IS CORRECTLY LOADED WITH THE BYTE
                         ;THAT RXOUT IS POINTING TO

RXOUT := TX[SIGNED_16[RXOUT)+1); {INCREMENT RXOUT}
     LD     HL,[RXOUT]
     INC    HL
     LD     [RXOUT],HL ;RXOUT IS CORRECTLY INCREMENTED

TEMP2 := RXOUT^;  {TEMP2 SHOULD GET THE NEXT BYTE}
     LD     [TEMP2],A  ;SINCE A WAS NOT DISTURBED, THE COMPILER DOES
                       ;NOT REALIZE THAT THE POINTER WAS UPDATED.
```

Temporary solution:
Set $AMNESIA ON$ around the pointer referencing code.

Signed off 04/29/87 in release 301.70

Number: D200036855 Product: 8085 B PASCAL VAX 64825S003 01.20

Keywords: IF

One-line description:
IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}

Problem:

```
VAR B1, B2 : BYTE;

BEGIN
IF B1 (>|<|=|<=|>=) B2 THEN
B1 := B1 - 1;   {THE REGISTER CONTAINING B1 IS DECREMENTED, THEN
                 OVERWRITTEN BEFORE IT IS SAVED IN MEMORY}
```

Temporary solution:

```
$AMNESIA +$
```

Signed off 04/29/87 in release 301.70

Number: D200040147 Product: 8085 B PASCAL VAX 64825S003 01.20

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for adding one char to another.

Problem:

```
VAR
SRC, DEST : CHAR;

BEGIN
DEST := DEST + SRC;  {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 04/29/87 in release 301.70

Number: D200058677 Product: 8085 B PASCAL VAX 64825S003 01.50

One-line description:
Using char and int. in control loop causes incorrect code to be gen'ed.

Problem:
If you use an integer and a char for the loop counters in a for loop incorrect code will be generated.

```
"processor"
$EXTENSIONS ON$
PROGRAM DOWNTO;
VAR
   I  :  SIGNED_16;
   N  :  SIGNED_16;
   J  :  BYTE;
   A  :  ARRAY[1..10] OF SIGNED_16;

BEGIN
   J:= 11;
   N:= 0;
   FOR I:= J-1 DOWNTO 1  DO        /* I is initialized incorrectly with
      BEGIN                           an undefined by DOWNTO+019H */
          A[I]:=I;
```

```
        N:= N+1;
    END;
END.
```

Temporary solution:
Declare the initializing variable (J in this example) to be of the same type as the index (I in this example).

Signed off 04/29/87 in release 301.70

Number: D200059659 Product: 8085 B PASCAL VAX 64825S003 01.50

One-line description:
$Range ON$ causes incorrect code to be generated for a test operation.

Problem:
The following program when compiled with the $RANGE ON$ option wil cause incorrect code to be generated.

```
"B8085" | "BZ80"
$EXTENSIONS$
$RANGE ON$

PROGRAM  BOOLREAL;

VAR  A,B,C    :   REAL;
     L        :   BOOLEAN;

BEGIN
     A := 10.0;
     B := 15.0;
     C := 12.0;


     L := (C < (B+.5)) AND ((C + .5) > A);
END.
```

The two intermediate results "(C < (B +.5))" and "((C+.5) >A)" are anded together and this result is compared with the value two. Thus the case is never true. With RANGE OFF correct code is generated.

Temporary solution:
It is necessary to turn $RANGE OFF$ to obtain correct code. Simply breaking up the expression will not work.

Signed off 04/29/87 in release 301.70

Number: D200060244 Product: 8085 B PASCAL VAX 64825S003 01.50

One-line description:
Incorrect data offsets in listing file.

Problem:
"processor name"
PROGRAM PROVE;


```
VAR
  X,Y:INTEGER;
  A: ARRAY[0..99999] OF INTEGER;
BEGIN
$TESTS 1, LIST_CODE ON, LIST_OBJ ON$
(* Comment ON
   Y := A[0];
   Y := A[8000];
   Y := A[9000];
   Comment OFF   *)
   $TESTS 3$
   Y := A[16000];
   Y := A[17000];
   $TESTS 7$
   Y := A[16000];
   Y := A[17000];
   $TESTS 1$
(* Comment ON
   Y := A[32000];
   Y := A[33000];
   Comment OFF   *)
END.
```

Temporary solution:
If arrays of this size are required download the file to the 64100 and compile.

Signed off 04/29/87 in release 301.70

Number: D200063925 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
functional type change of a constant into multi-byte structure gen's err

Problem:
Functional type casting of a constant into a multi-byte structure generates bad data.

```
"processor"

PROGRAM BAD_DATA;

TYPE  EVENT = RECORD
        A   : BYTE;
        B   : BYTE;
        C   : INTEGER;
        D   : BYTE;
      END;

VAR   EVENT1   : EVENT;


PROCEDURE   GENERATOR();
   BEGIN
      EVENT1 := EVENT(0);  { THIS ASSIGNMENT RESULTS IN BAD DATA }
   END;
```

```
BEGIN
END.
```

Signed off 04/29/87 in release 301.70

Number: D200064345 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
Error #1009 using byte-sized ORG'ed variables in FOR loops

Problem:
Error #1009 is generated when byte sized ORG'ed variables are used in FOR loops. The following code illustrates the problem.

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
PROCEDURE ERR;
VAR
$ORG 5000$
   B1,B2,X1: BYTE;

BEGIN

   FOR X1 := B1 to B2 DO;      (*Pass 2 Error 1009 - No free registers*)
END;
.
```

Temporary solution:
The error does not occur if the FOR loop variable is word sized instead of byte sized. It will also go away if the ORG statement is removed.

Signed off 04/29/87 in release 301.70

Number: D200064444 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
32-bit unsigned divide and modulus may fail

Problem:
The result of an unsigned 32-bit division or modulus operation may be incorrect if the dividend and the destination are the same location. The problem is in the library routine Zdworddiv. The following code demonstrates the problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2 : UNSIGNED_32;
BEGIN
  B1 := UNSIGNED_32(0E00000000);
  B2 := UNSIGNED_32(0900000000);
  B1 := B1/B2;
END.
```


Signed off 04/29/87 in release 301.70

Number: D200064519 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
Library routine REAL_ROUND may fail.

Problem:
The library routine REAL_ROUND may fail, causing floating point numbers to be incorrectly rounded to integers.

Signed off 04/29/87 in release 301.70

Number: D200064584 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
DEBUG byte division and modulus may incorrectly report division by zero

Problem:
The DEBUG library routines for performing signed and unsigned byte division and modulus operations may fail and incorrectly report an attempted division by zero.

The following code fails in this manner:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2,B3 : BYTE;
$ORG 5000H$
  BA : ARRAY[1..15] OF BYTE;

BEGIN
  B1 := 1;
  B2 := 1;
  B3 := 0;
  BA[B3] := B1 DIV B2;   (*DIV fails - reports division by zero*)
END.
```

Signed off 04/29/87 in release 301.70

Number: D200064972 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
Set comparisons with the empty set may fail

Problem:
Set comparisons with the empty set may fail. The following code is an example of this problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
TYPE
  CH = 0..127;
  SET1 = SET OF CH;
```

```
VAR
  S1 : SET1;
PROCEDURE ERROR; EXTERNAL;
BEGIN
  S1 := [];
  IF S1 <> [] THEN          (*In CONST_prog, not enough bytes are
    ERROR;                    defined for the set*)
END.
```

Signed off 04/29/87 in release 301.70

Number: D200065342 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
Assignment of constant string of length 1 to string variable may fail.

Problem:
Assignment of a constant string of length 1 to a string variable that is itself a multidimensional array element may fail.

First, the address of the destination string is calculated in HL. Then the value of the string length resulting from the assignment, i.e. one (1), is loaded into the position reserved for the length of the string via a store indirect through HL. Up to this point all is as it should be; however, the value of the single character that comprises the string is then also stored HL indirect, overwriting the length and failing to correctly load the string value. The HL register should be incremented before the second store.

The following is an example:

```
"processor name"
PROGRAM TEST;
TYPE
  STRING_15 = PACKED ARRAY[0..15] OF CHAR;
VAR
  TWO_D_ARR : ARRAY[1..3,1..3] OF STRING_15;
BEGIN
  TWO_D_ARR[2,1] := " ";
        LD   HL,0030H
        PUSH HL
        LD   HL,00002H
        PUSH HL
        LD   HL,00010H
        PUSH HL
        LD   HL,00001H
        PUSH HL
        LD   BC,DTEST-00040H
        LD   A,002H
        CALL Zarrayref
        LD   A,001H
        LD   [HL],A      (*or LD   M,A  *)
        LD   A,020H
        LD   [HL],A       (*This is the error - should INC HL first*)
END.
```

Signed off 04/29/87 in release 301.70

Number: D200067447 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
Assignment of unsigned_8 variables to expression always assigns zero.

Problem:
The following example program generates incorrect code:

```
"processor name"

PROGRAM TEST;

VAR
  CP,DP,TP,SBS : UNSIGNED_8;

BEGIN
  CP:=TP;
  IF (CP > DP) THEN
     SBS:= CP - DP;   (*This always assigns zero to SBS*)
END.
```

Signed off 04/29/87 in release 301.70

Number: D200067017 Product: 8085 B PASCAL VAX 64825S003 01.60

One-line description:
.A,.R, and listing files should reside in directory compile is executed.

Problem:
The output files from the compilers and assemblers (.A and .R) reside in the directory that the source file being operated on resides in. This was a change implemented in the October 1986 SMS in order to make the VAX hosted compilers consistent with the HP-UX hosted compilers. Problems occur if the source file resides in a read-only directory. Customer feels that the resultant files from the compile should reside in the directory the compiler is invoked from.

Signed off 04/29/87 in release 301.70

Number: D200011262 Product: 8085 C 64826 01.00

Keywords: PASS 1

One-line description:
Functions invoked via function pointers may JSR the wrong location.

Problem:
When the typedef statement is used to define pointers to functions, and this pointer type is used in a cast of a variable array to invoke code stored in that array, program execution may transfer to the wrong location. For example, in the following code the simple call to code_array fails while the call and assignment to p works correctly:

```
typedef int(*PFI)();  /* PFI a pointer to int functions */
int code_array[100];  /* array contains code */
PFI p;                /* p a pointer of type PFI */

pfibug()
{
  (*((PFI) code_array))(); /* fails in JSR to code_array */
  (*(p=(PFI)code_array))(); /* assignment and JSR successful */
}
```

Temporary solution:
Set up a dummy variable and perform an assignment to it when doing this type of operation.

Signed off 04/29/87 in release 601.04

Number: D200011346 Product: 8085 C 64826 01.00

Keywords: PASS 1

One-line description:
Unsigned integers treated as signed when subtracted from pointers.

Problem:
When an unsigned short or integer is used as an offset to a pointer, the unsigned will be treated as a signed when doing pointer calculations. Offsets large enough to set the sign bit will be interpreted as a negative offset when the offset is subtracted from a pointer. The following code exibits the problem if offset is greater than 32767 dec.

```
unsigned offset;
struct { int a,b,c;
       } *ptr;
unsigned long x;

main ()
{
  x = ptr - offset;  /* The compiler will generate code negating   */
}                    /* offset for the "-" operation.              */
```

Temporary solution:
Cast the offset in the expression as the next larger integer.
ie. x = ptr - (unsigned long)offset;

Signed off 04/29/87 in release 601.04

Number: D200025742 Product: 8085 C 64826 01.01

Keywords: CODE GENERATOR

One-line description:
Assigning a ptr. after its post incr/decr. gives incorrect value.

Problem:
Pointer assignment after a post increment or decrement to that pointer stores incorrect value. The following is an illustration:

```
"C"
"PROCESSOR_NAME"

unsigned short fct(g)
unsigned short *g;
{
   unsigned short a,b;
   b=*g;
   *g++;
   a=*g;
}
```

The first assignment statement stores the contents of what g is point-to in the accumulator. Once the pointer is incremented, the compiler loads the accumulator (which still has the previous value) into the variable a. The compiler is false remembering the value in the accumulator as the current contents of what g is pointing to.

Temporary solution:
Turn $AMNESIA ON$ to force the reload of the accumulator from the BC register pair.

Signed off 04/29/87 in release 601.04

Number: D200037622 Product: 8085 C 64826 01.01

One-line description:
IF statements involving return values and address calculations may fail.

Problem:
HP9000 compiler generates different code from 64000 and VAX, and both are wrong. If an if statement compares the value returned from a function with a value obtained via the structure pointer operator, the value returned from the function may be overwritten by the address of the structure element. This will cause the test to be erroneous.

Example:

```
"C"
"8085"

extern unsigned x();
struct
{long *ptr;
```

```
unsigned length;
} *now_string;

func_1()
 {
if(x() < now_string->length) /* test fails */
return(5);
 }
```

Temporary solution:
Use a temporary variable to hold the return result of the function.

Signed off 04/29/87 in release 601.04

Number: D200040444 Product: 8085 C 64826 01.01

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                      case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 601.04

Number: D200059907 Product: 8085 C 64826 01.02

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 601.04

Number: D200062828 Product: 8085 C 64826 01.02

One-line description:
Incorrect code generated when function parameter is post incremented.

Problem:
Incorrect code is generated when a function argument is post incremented. The following code is an example of this problem:

```
"C"
"8085"
$RECURSIVE OFF$
void puts(buf);
char *buf;
{
  while (*buf != '\0')
       putchar(*buf++);
}
```

Temporary solution:
Pass the parameter to the function without post incrementing it, then increment it after the function call.

Signed off 04/29/87 in release 601.04

Number: D200063297 Product: 8085 C 64826 01.03

Keywords: CODE GENERATOR

One-line description:
Character isn't converted to int before calculations use it

Problem:
Kernigan and Ritchie states that a character is converted to an integer before calculations use the char variable. Our compiler does not conver t the character to an integer prior to any calculations.

For example:

```
"C"
"8086"
main() {
  char c;
  int i;
  i = ((c<< 4) *5)/i;
```

AX register if c = 0FFH

```
xxxx     MOV  CL,#+00004H   {moves 4 into counter}
00xx     MOV  AH,#0         {00h into AH}
00FF     MOV  AL,SS:BYTE PTR[BP-00003H]  {loads c into AL}
00F0     SHL  AL,CL       {shifts left 4 c ;however, it loses the uppe
                           r byte because it was not SHL AX,CL}
 }
```

The character is not being treated as an integer. Making this SHL AX,CL would fix the problem.

Emulating the generated code confirmed that the high byte (4 places) was not being shifted into AH.

Temporary solution:
Type cast c to be an integer before using it in the expression.

Signed off 04/29/87 in release 601.04

Number: D200063610 Product: 8085 C 64826 01.03

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 601.04

Number: D200064881 Product: 8085 C 64826 01.03

One-line description:
Funct calls via pointers with parms cause subsequent stack ref errors

Problem:

When functions are called via pointers and are passed parameters, subsequent references to stack relative objects will be incorrect. The following code is an example of this problem:

```
"C"
"processor name"
extern int called_func();
typedef int (*PFI)();
PFI call_ptr = called_func;

main()
{
  int local;

  local = 6;              (*variable is accessed correctly*)
  (*(call_ptr() (1,2);    (*function call via pointer with parameters*)
  local = 3;              (*wrong location accessed*)
}
```

Signed off 04/29/87 in release 601.04

Number: D200066209 Product: 8085 C 64826 01.03

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 601.04

Number: D200051979 Product: 8085 C 300 64826S004 01.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"processor name"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
       int  c;
            switch(c){
                 case1:         break;
                 default:       switch(c){
                                   case 2: break;
                                }
                                break;
            }
}
```

Signed off 04/29/87 in release 401.20

Number: D200059931 Product: 8085 C 300 64826S004 01.00

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"
```

```
main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 401.20

Number: D200063321 Product: 8085 C 300 64826S004 01.10

Keywords: CODE GENERATOR

One-line description:
Character isn't converted to int before calculations use it

Problem:
Kernigan and Ritchie states that a character is converted to an integer before calculations use the char variable. Our compiler does not conver t the character to an integer prior to any calculations.

For example:

```
   "C"
   "8086"
   main() {
     char c;
     int i;
     i = ((c<< 4) *5)/i;

 AX register if c = 0FFH
 ------
  xxxx     MOV  CL,#+00004H   {moves 4 into counter}
  00xx     MOV  AH,#0         {00h into AH}
  00FF     MOV  AL,SS:BYTE PTR[BP-00003H]  {loads c into AL}
  00F0     SHL  AL,CL       {shifts left 4 c ;however, it loses the uppe
                              r byte because it was not SHL AX,CL}
   }
```

The character is not being treated as an integer. Making this SHL AX,CL would fix the problem.

Emulating the generated code confirmed that the high byte (4 places) was not being shifted into AH.

Temporary solution:
Type cast c to be an integer before using it in the expression.

Signed off 04/29/87 in release 401.20

Number: D200063644 Product: 8085 C 300 64826S004 01.10

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 401.20

Number: D200064915 Product: 8085 C 300 64826S004 01.10

One-line description:
Funct calls via pointers with parms cause subsequent stack ref errors

Problem:
When functions are called via pointers and are passed parameters, subsequent references to stack relative objects will be incorrect. The following code is an example of this problem:

```
"C"
"processor name"
extern int called_func();
typedef int (*PFI)();
PFI call_ptr = called_func;

main()
{
  int local;

  local = 6;              (*variable is accessed correctly*)
  (*(call_ptr() (1,2);    (*function call via pointer with parameters*)
  local = 3;              (*wrong location accessed*)
}
```

Signed off 04/29/87 in release 401.20

Number: D200066233 Product: 8085 C 300 64826S004 01.10

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
```

```
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 401.20

Number: D200066555 Product: 8085 C 300 64826S004 01.10

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 401.20

Number: 5000179028 Product: 8085 C 500 64826S001 01.05

One-line description:
Number of errors listed at bottom of the listing is incorrect.

Problem:
The following program, compiled with the -o option, does not correctly total the number of errors at the end of the listing

```
"C"
"8085"
 int x,y;
 main ()
  { a ++; x = val;  v =0;
    funct1(x,y);
    if (func2());
    if (var == 0);
  }
```

If the line "if (var == 0);" is removed, the problem goes away. It appears only to happen when an undefined variable is used in an if statement.

A shorter example,

```
      "C"
      "8085"
       main () { if (var == 0); }
                          ^103,104^407
103:
104: etc.
407: etc.

End of compilation, number of errors= 0
```

Temporary solution:
Refer to the listing generated to determine errors.

Signed off 04/29/87 in release 101.60

Number: D200025759 Product: 8085 C 500 64826S001 01.10

Keywords: CODE GENERATOR

One-line description:
Assigning a ptr. after its post incr/decr. gives incorrect value.

Problem:
Pointer assignment after a post increment or decrement to that pointer stores incorrect value. The following is an illustration:

```
"C"
"PROCESSOR_NAME"

unsigned short fct(g)
unsigned short *g;
{
```

```
   unsigned short a,b;
   b=*g;
   *g++;
   a=*g;
}
```

The first assignment statement stores the contents of what g is point-
to in the accumulator. Once the pointer is incremented, the compiler
loads the accumulator (which still has the previous value) into the
variable a. The compiler is false remembering the value in the
accumulator as the current contents of what g is pointing to.

Temporary solution:
Turn $AMNESIA ON$ to force the reload of the accumulator from the BC
register pair.

Signed off 04/29/87 in release 101.60

Number: D200040451 Product: 8085 C 500 64826S001 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner
switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
```



```
}
```

Signed off 04/29/87 in release 101.60

Number: D200042085 Product: 8085 C 500 64826S001 01.20

One-line description:
IF statements involving return values and address calculations may fail.

Problem:
HP9000 compiler generates different code from 64000 and VAX, and both
are wrong. If an if statement compares the value returned from
a function with a value obtained via the structure pointer operator,
the value returned from the function may be overwritten by the address
of the structure element. This will cause the test to be erroneous.

Example:

```
"C"
"8085"

extern unsigned x();
struct
{long *ptr;
unsigned length;
} *now_string;

func_1()
 {
if(x() < now_string->length) /* test fails */
return(5);
 }
```

Temporary solution:
Use a temporary variable to hold the return result of the function.

Signed off 04/29/87 in release 101.60

Number: D200059915 Product: 8085 C 500 64826S001 01.40

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the
compiler to hang. I could not duplicate this problem, but, the
compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 101.60

Number: D200063305 Product: 8085 C 500 64826S001 01.50

Keywords: CODE GENERATOR

One-line description:
Character isn't converted to int before calculations use it

Problem:
Kernigan and Ritchie states that a character is converted to an integer before calculations use the char variable. Our compiler does not conver t the character to an integer prior to any calculations.

For example:

```
"C"
"8086"
main() {
  char c;
  int i;
  i = ((c<< 4) *5)/i;
```

```
AX register if c = 0FFH
------
 xxxx     MOV  CL,#+00004H   {moves 4 into counter}
 00xx     MOV  AH,#0         {00h into AH}
 00FF     MOV  AL,SS:BYTE PTR[BP-00003H]  {loads c into AL}
 00F0     SHL  AL,CL       {shifts left 4 c ;however, it loses the uppe
                            r byte because it was not SHL AX,CL}
  }
```

The character is not being treated as an integer. Making this SHL AX,CL would fix the problem.

Emulating the generated code confirmed that the high byte (4 places) was not being shifted into AH.

Temporary solution:
Type cast c to be an integer before using it in the expression.

Signed off 04/29/87 in release 101.60

Number: D200063628 Product: 8085 C 500 64826S001 01.50

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 101.60

Number: D200064899 Product: 8085 C 500 64826S001 01.50

One-line description:
Funct calls via pointers with parms cause subsequent stack ref errors

Problem:
When functions are called via pointers and are passed parameters, subsequent references to stack relative objects will be incorrect. The following code is an example of this problem:

```
"C"
"processor name"
extern int called_func();
typedef int (*PFI)();
PFI call_ptr = called_func;

main()
{
  int local;

  local = 6;              (*variable is accessed correctly*)
  (*(call_ptr() (1,2);    (*function call via pointer with parameters*)
  local = 3;              (*wrong location accessed*)
}
```

Signed off 04/29/87 in release 101.60

Number: D200066217 Product: 8085 C 500 64826S001 01.50

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"
```

```
char  string[]  =  "do it this way";

main()
{
  int i; ·

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 101.60

Number: D200066530 Product: 8085 C 500 64826S001 01.50

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 101.60

Number: D200025767 Product: 8085 C VAX 64826S003 01.10

Keywords: CODE GENERATOR

One-line description:
Assigning a ptr. after its post incr/decr. gives incorrect value.

Problem:
Pointer assignment after a post increment or decrement to that pointer stores incorrect value. The following is an illustration:

```
"C"
"PROCESSOR_NAME"

unsigned short fct(g)
unsigned short *g;
{
   unsigned short a,b;
   b=*g;
   *g++;
   a=*g;
}
```

The first assignment statement stores the contents of what g is point-to in the accumulator. Once the pointer is incremented, the compiler loads the accumulator (which still has the previous value) into the variable a. The compiler is false remembering the value in the accumulator as the current contents of what g is pointing to.

Temporary solution:
Turn $AMNESIA ON$ to force the reload of the accumulator from the BC register pair.

Signed off 04/29/87 in release 301.90

Number: D200040469 Product: 8085 C VAX 64826S003 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
```

```
                        evaluating 'case 1' above.  */
            }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
                switch(c){
                     case1:         break;
                     default:       switch(c){
                                       case 2: break;
                                    }
                                    break;
                }
}
```

Signed off 04/29/87 in release 301.90

Number: D200042093 Product: 8085 C VAX 64826S003 01.20

One-line description:
IF statements involving return values and address calculations may fail.

Problem:
HP9000 compiler generates different code from 64000 and VAX, and both are wrong. If an if statement compares the value returned from a function with a value obtained via the structure pointer operator, the value returned from the function may be overwritten by the address of the structure element. This will cause the test to be erroneous.

Example:

```
"C"
"8085"

extern unsigned x();
struct
{long *ptr;
unsigned length;
} *now_string;

func_1()
 {
if(x() < now_string->length) /* test fails */
return(5);
 }
```

Temporary solution:
Use a temporary variable to hold the return result of the function.

Signed off 04/29/87 in release 301.90

Number: D200059923 Product: 8085 C VAX 64826S003 01.60

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 301.90

Number: D200063313 Product: 8085 C VAX 64826S003 01.80

Keywords: CODE GENERATOR

One-line description:
Character isn't converted to int before calculations use it

Problem:
Kernigan and Ritchie states that a character is converted to an integer before calculations use the char variable. Our compiler does not conver t the character to an integer prior to any calculations.

For example:

```
   "C"
   "8086"
   main() {
     char c;
     int i;
     i = ((c<< 4) *5)/i;

 AX register if c = 0FFH
 ------
  xxxx     MOV  CL,#+00004H   {moves 4 into counter}
  00xx     MOV  AH,#0         {00h into AH}
  00FF     MOV  AL,SS:BYTE PTR[BP-00003H]  {loads c into AL}
  00F0     SHL  AL,CL       {shifts left 4 c ;however, it loses the uppe
                              r byte because it was not SHL AX,CL}
   }
```

The character is not being treated as an integer. Making this SHL AX,CL would fix the problem.

Emulating the generated code confirmed that the high byte (4 places) was not being shifted into AH.

Temporary solution:
Type cast c to be an integer before using it in the expression.

Signed off 04/29/87 in release 301.90

Number: D200063636 Product: 8085 C VAX 64826S003 01.80

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 301.90

Number: D200064907 Product: 8085 C VAX 64826S003 01.80

One-line description:
Funct calls via pointers with parms cause subsequent stack ref errors

Problem:
When functions are called via pointers and are passed parameters, subsequent references to stack relative objects will be incorrect. The following code is an example of this problem:

```
"C"
"processor name"
extern int called_func();
typedef int (*PFI)();
PFI call_ptr = called_func;

main()
{
  int local;

  local = 6;              (*variable is accessed correctly*)
  (*(call_ptr() (1,2);    (*function call via pointer with parameters*)
  local = 3;              (*wrong location accessed*)
}
```

Signed off 04/29/87 in release 301.90

Number: D200066225 Product: 8085 C VAX 64826S003 01.80

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"
```



```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 301.90

Number: D200066548 Product: 8085 C VAX 64826S003 01.80

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 301.90

Number: 5000136093 Product: 8086/8 ASSEMB 64853 02.00

Keywords: CODE GENERATOR

One-line description:
Index addressing in MOV statement creates incorrect code

Problem:
The following program generates incorrect code:

```
        "80186"
        NNN     EQU     0EEH
                MOV     AL,NNN                  ;line A
                MOV     AL,ES:BYTE PTR NNN[BX]  ;line B
                MOV     AL,ES:BYTE PTR [NNN+BX] ;line C
                MOV     AL,ES:BYTE PTR [NNN][BX] ;line D
```

Line A generates B0EE which is correct; and line C and line D both generate 268A87EE00 which is also correct. As expected these two lines generate the same code. Line B, however, generates B0EE (same as line A) which appears incorrect. Line B can be interpreted as: the BYTE PTR of (the offset of NNN from ES + the contents of BX) should be moved into AL. This does not seem to be happening since the insertion of the statement MOV BX,#01H before line B does not change the value being moved into AL in line B (still generates B0EE).

Temporary solution:
No known temporary solution.

Signed off 04/29/87 in release 302.02

Number: 5000136226 Product: B0B6/B ASSEMB 64B53 02.00

One-line description:
Corrupt file generated by assem. when large # of files are link. w/xref

Problem:
This problem can be demonstrated by using a customer tape that is available from Robin Barker-Chambers. This only occurs when generating a xref. Softfix must be run to resolve the problem. The absolute file generated is always correct.

Signed off 04/29/87 in release 302.02

Number: 5000152090 Product: B0B6/B ASSEMB 64B53 02.01

One-line description:
Assembler does not flag LR error when short jump > +/- 127 bytes

Problem:
When a short jump is used as follows:

```
  "801B6"
    JMP SHORT LABEL
    DBS     B1H
  LABEL
```

This program does not generate a LR error (illegal range error) even

though the jump ois greater than +/-127 bytes. According to Intel a short jump is only allowed a +/-127 byte range.

Temporary solution:
No known temporary solution.

Signed off 04/29/87 in release 302.02

Number: 5000154542 Product: B0B6/B ASSEMB 64B53 02.01

One-line description:
OLD_8087 directive is ignored after the use of DQ pseudo

Problem:
The "OLD_8087" directive is ignored following the use of the pseudo DQ.

```
  "8086"
        OLD_8087
        DQ      1
        FDIVRP ST[1],ST     generates 9BDEF1 -- wrong
        OLD_8087
        FDIVRP ST[1],ST     generates 9BDEF9 -- correct
        END
```

The DQ cancells the OLD_B0B7 directive.

Temporary solution:
Use OLD_B0B7 pseudo after using DQ pseudo.

Signed off 04/29/B7 in release 302.02

Number: 5000161B36 Product: 80B6/B ASSEMB 64B53 02.01

One-line description:
FMUL ST[3],ST[5] does not flag error

Problem:
8086 assembler does not generate an error for the following illegal instruction:

```
       FMUL     ST[3],ST[5]
```

Code for FMUL ST[3],ST is generated.

```
FMUL  ST,ST      generates ---> 9BDCC8     (expected 9BDBCB)
FMUL  ST[0],ST   generates ---> 9BDCCB     (expected 9BD8C8)
FMUL  ST,ST[0]   generates ---> 9BD8CB
```

All three of these statements is equivalent
The two different opcodes generated do the same thing.

Temporary solution:
No known temporary solution.

Signed off 04/29/87 in release 302.02

Number: D200005116 Product: 8086/8 ASSEMB 64853 00.08

Keywords: LINKER

One-line description:
"Total # of bytes loaded" is incorrect if segment boundary is crossed.

Problem:
The linker listing indicates an incorrect number of bytes being loaded when a segment boundary is crossed.

```
     ex.      "8086"
                ORG    00FFF0H
                DB     0H,1H,2H,3H,4H,5H,6H,7H,8H,9H
                DB     0AH,0BH,0CH,0DH,0EH,0FH
                PROG
                NOP
                END
```

Causes the following Linker listing;

```
 FILE/PROG NAME      PROGRAM   DATA   COMMON  ABSOLUTE          DATE...
-------------------------------------------------------------------------
NAME:UID             00000000                0000FFF0-FFFFFFFF
next address         00000001

XFER address= 00000000 Defined by DEFAULT
absolute & link_com file name=NAME:UID
Total# of bytes loaded= FFFF0011
```

Temporary solution:
Avoid crossing segment boundaries at link time.

Signed off 04/29/87 in release 302.02

Number: D200033563 Product: 8086/8 ASSEMB 64853 02.00

One-line description:
STACKSEG pseudo op does not allocate space correctly.

Problem:
The STACKSEG pseudo instruction should allow the user to create a logical stack segment of a specified length in bytes. Instead, the assembler creates a segment where the number of bytes allocated is exactly the value of the current program counter.

Signed off 04/29/87 in release 302.02

Number: D200042242 Product: 8086/8 ASSEMB 64853 02.00

One-line description:
Expression type errors occur for legal INC instructions.

Problem:
The following code generates errors as shown when assembled, although each of the instructions are legal.

```
"processor name"
          ASSUME      CS:PROG
          PROG
          INC         [BX][SI]
                      ^ET (Expression type)
          INC         [BX+SI]
                            ^ET,IE (Illegal expression)
```


Temporary solution:
Do not use INC instructions of this form.

Signed off 04/29/87 in release 302.02

Number: D200043885 Product: 8086/8 ASSEMB 64853 02.00

One-line description:
Macro called with more parameters than declared generates error.

Signed off 04/29/87 in release 302.02

Number: D200052100 Product: 8086/8 ASSEMB 300 64853S004 02.00

One-line description:
Expression type errors occur for legal INC instructions.

Problem:
The following code generates errors as shown when assembled, although each of the instructions are legal.

```
"processor name"
        ASSUME      CS:PROG
        PROG
        INC         [BX][SI]
                    ^ET (Expression type)
        INC         [BX+SI]
                          ^ET,IE (Illegal expression)
```

Temporary solution:
Do not use INC instructions of this form.

Signed off 04/29/87 in release 402.20

Number: D200042556 Product: 8086/8 ASSEMB 500 64853S001 02.00

One-line description:
Expression type errors occur for legal INC instructions.

Problem:
The following code generates errors as shown when assembled, although each of the instructions are legal.

```
"processor name"
        ASSUME      CS:PROG
        PROG
        INC         [BX][SI]
                    ^ET (Expression type)
        INC         [BX+SI]
                          ^ET,IE (Illegal expression)
```

Temporary solution:
Do not use INC instructions of this form.

Signed off 04/29/87 in release 102.30

Number: D200042564 Product: 8086/8 ASSEMB VAX 64853S003 02.00

One-line description:
Expression type errors occur for legal INC instructions.

Problem:
The following code generates errors as shown when assembled, although each of the instructions are legal.

```
"processor name"
          ASSUME      CS:PROG
          PROG
          INC         [BX][SI]
                      ^ET (Expression type)
          INC         [BX+SI]
                            ^ET,IE (Illegal expression)
```

Temporary solution:
Do not use INC instructions of this form.

Signed off 04/29/87 in release 302.40

Number: 5000108969 Product: 8086/8 C 64818 02.00

One-line description:
Dereferencing a structue is not working properly.

Temporary solution:
Use the alternate dereferencing structure (many.one) suggested above.

Signed off 04/29/87 in release 803.02

Number: 5000135913 Product: 8086/8 C 64818 02.00

Keywords: CODE GENERATOR

One-line description:
AX not loaded with constant prior to using it to calculate expression

Problem:

```
"C"
"8086"
    char a[2][2];
    char b;
    char c[8];
 bug() { char i,j,k;
         b=a[j][k];
         c[i*2] =b;
            MOV AL,#+00002H          (AL CONTAINS 2H)
            MUL SS:BYTE PTR [BP-00003H]   (AX CONTAINS i*2)
                 :
         b = c[i*2+1];  }
            MUL SS:BYTE PTR [BP-00003H]   (AX CONTAINS i*i*2 -WRONG)
                                          (THIS OCCURED BECAUSE ax WAS
                                           ASSUMED TO CONTAIN 2H)
```

Temporary solution:
NO KNOWN TEMPORARY SOLUTION

Signed off 04/29/87 in release 803.02

Number: 5000160770 Product: 8086/8 C 64818 02.00

Keywords: CODE GENERATOR

One-line description:
The compiler generates incorrect code for floating point constants

Problem:
The data generated by the compiler for floating point constants is not always correct. This problem does not occur on the 64100.

For example,

```
  "C"
  "8086"
                              Assembly code generated
  float var1 = 32.0;           DW 0000H, 4200H
  float var2 = 32/1;           DW 0000H, 4200H
  float var3 = 32.0/1;         DW 0000H, 4280H
```

```
  float var4 = 32/1.0;        DW 0000H, 4280H
  float var5 = 32.0/1.0;      DW 0000h, 4280H
```

All of these expressions should generate: DW 0000H,4200H

Temporary solution:
The temporary solution is:

(1) Compile on the 64100.

or

(2) Expressions with constant operand(s) that are floating point numbers may not be initialized in the declarations. They should be initialized in the main program and type cast.

For example,

```
float  var1;

main () {
          var1 = (float)32/1.0;
          var1 = 32.0/(float)1;
        }
```

Signed off 04/29/87 in release 803.02

Number: D200007831 Product: 8086/8 C 64818 00.56

Keywords: CODE GENERATOR

One-line description:
Error #1006 generated when incorrect value returned from a function

Problem:
When a function is declared to return one type of value, but another type is actually returned, the compiler may generate Pass 2 error #1006. The following code exhibits this problem.

```
"C"
"8086"

char *char_ptr()
{
  int i;
  return((long)i);
}
```

Temporary solution:
Have the return statement in each function send back a value that matches the type declaration of the function.

Signed off 04/29/87 in release 803.02

Number: D200010116 Product: 8086/8 C 64818 00.56

Keywords: PASS 1

One-line description:
Unsigned integers treated as signed when subtracted from pointers.

Problem:
When an unsigned short or integer is used as an offset to a pointer, the unsigned will be treated as a signed when doing pointer calculations. Offsets large enough to set the sign bit will be interpreted as a negative offset when the offset is subtracted from a pointer. The following code exibits the problem if offset is greater than 32767 dec.

```
unsigned offset;
struct { int a,b,c;
       } *ptr;
unsigned long x;

main ()
{
  x = ptr - offset;  /* The compiler will generate code negating   */
}                    /* offset for the "-" operation.              */
```

Temporary solution:
Cast the offset in the expression as the next larger integer.
ie. x = ptr - (unsigned long)offset;

Signed off 04/29/87 in release 803.02

Number: D200011395 Product: 8086/8 C 64818 00.56

Keywords: PASS 1

One-line description:
Functions invoked via function pointers may JSR the wrong location.

Problem:
When the typedef statement is used to define pointers to functions, and this pointer type is used in a cast of a variable array to invoke code stored in that array, program execution may transfer to the wrong location. For example, in the following code the simple call to code_array fails while the call and assignment to p works correctly:

```
typedef int(*PFI)();  /* PFI a pointer to int functions */
int code_array[100];  /* array contains code */
PFI p;                /* p a pointer of type PFI */

pfibug()
{
  (*((PFI) code_array))();  /* fails in JSR to code_array */
  (*(p=(PFI)code_array))(); /* assignment and JSR successful */
}
```

Temporary solution:
Set up a dummy variable and perform an assignment to it when doing this type of operation.

Signed off 04/29/87 in release 803.02

Number: D200040295 Product: 8086/8 C 64818 02.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                    case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 803.02

Number: D200059675 Product: 8086/8 C 64818 03.00

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"
```



```
main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 803.02

Number: D200063388 Product: 8086/8 C 64818 03.01

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 803.02

Number: D200065979 Product: 8086/8 C 64818 03.01

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
```

```
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 803.02

Number: D200049973 Product: 8086/8 C 300 64818S004 03.00

One-line description:
With $POINTER_SIZE 32$ assigning an address + a sizeof in 1 line fails.

Problem:
The following code illustrates the problem:

```
"C"
"80186"
$POINTER_SIZE 32$
char *ptr;
fct(ptr);
{
  char *sptr;
  sptr = &ptr + sizeof(char *);
  sptr = &ptr;
  sptr += sizeof(char *);
}
```

Different data is assigned to the pointer if the assignment is written in one or two lines. The difference between both the loaded addresses should be 4.

Signed off 04/29/87 in release 403.20

Number: D200051912 Product: 8086/8 C 300 64818S004 03.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"processor name"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"processor name"

main(){
       int  c;
            switch(c){
```

```
                    case1:          break;
                    default:        switch(c){
                                       case 2: break;
                                    }
                                    break;
                }
}
```

Signed off 04/29/87 in release 403.20

Number: D200059709 Product: 8086/8 C 300 64818S004 03.00

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 403.20

Number: D200063412 Product: 8086/8 C 300 64818S004 03.10

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 403.20

Number: D200066001 Product: 8086/8 C 300 64818S004 03.10

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"
```



```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 403.20

Number: D200066381 Product: 8086/8 C 300 64818S004 03.10

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 403.20

Number: 5000149773 Product: 8086/8 C 500 64818S001 03.00

One-line description:
Both operands of expression loaded into AX when calculating array index

Problem:
The following program loads both operands of an expression into
AX and then tries to add them together by adding AX to AX.

```
"C"
"80188"
$FAR_EXTVARS$
$POINTER_SIZE 32$
$FAR_PROC$

struct Button_Obj {
         char button_code;
         char label_code;
         char button_parm;
         char button_attrib;
         );

extern char Channel_Data[6][9];
#define Channel_Type  8
   extern struct Button_obj Current_Buttons[60];
   extern char S_D_Button_COdes[5][8];
   extern struct Button_obj Stat_Btns[];

Copy_Buttons( Table_Ptr, Table_Size)
   struct Button_obj (Table_ptr)[];
   char  Table_Size;
  { char counter;
    for (counteer=0; counteeer < Table_size; counter++)
        {  Current_Buttons[counter] = Table_ptr[counter];
        }
    return;
  )

Setup_stat()
   {
    char display_trace;
    char column;
    char btn_code;
    Copy_Buttons(Stat_Btns, 54);
    for (display_trace = 0; display_trace <= 5; display_trace++)
      {
      for (column =0; column < 8; column++)
        btn_code = S_D_Button_Codes[Channel_Data[display_trace]
                   [Channel_Type][column];
                   :
              MOV    AL,SS:BYTE PTR [BP - 00004H]
                   :   (loads column into AX)
        Current_Buttons[column + display_trace*8].button_code = (btn_cod
e & 0x7F);
              MOV AL,#+00008H
              MUL SS:BYTE PTR  [BP-00005H]   (loads display_trace*8
                                              into AX over column)
```



```
              ADD AX,AX      (tries to add display_trace*8 to column
                               not correct)
  }
}
return; }
```

Temporary solution:
Use the compiler option, $AMNESIA ON$, around the incorrect state-
ment. This will force the compiler to forget the register contents
after each statement.

Signed off 04/29/87 in release 103.30

Number: 5000152108 Product: 8086/8 C 500 64818S001 03.00

Keywords: CODE GENERATOR

One-line description:
ES registeris overwritten when loading a ptr. w/ addr.of a structure

Problem:
The following program overwrites the ES register when using two
levels of indirection.

```
"C"
"80188"
$FAR_EXTVARS$
$POINTER_SIZE 32$

struct Button_Def {
       char button_char;
       char next;
       char *(*labels)[];
       int  (*button_proc)();
       int  (*setup_proc)();
       };
struct Button_Obj {
       char butx1,buty1,butx2,buty2;
       char button_code;
       char label_code;
       char button_parm;
       char button_attrib;
       };

#define BCont  0x04
#define BAccel 0.08
#define REPEAT0  80
#define DELAY    160
#define ENTER    0x0000
#define RTimer   TP_Timer_Cnt

extern struct Button_Def Button_List[];
extern struct Button_Obj Current_Buttons[];
extern char UserState;
extern char RepeatInterval;
extern char TP_Timer_Cnt;
```

```
extern char cbutn;
extern char bchar;
extern Put_Button();

DebounceTimer()
{
   struct Button_Def *but_p;
   struct Button_Obj *obj_p;

   switch (UserState) {

   case 1:
      Put_Button(ENTER|(int)cbutn);
      Put_Button( 0x0000|(int)cbutn);
      UserState = 2;
      obj_p = &Current_Buttons[cbutn];
      but_p = &Button_List[obj_p->button_code];
       :
      MOV   AX,SEG Button_List
      MOV   ES,AX              ;ES contains segment of Button_List
       :
      LES   SI,SS:DWORD PTR [BP-6H]  ;ES contains seg. of obj_p
       :
      MOV   SS:WORD PTR[BP-8H],ES  ; moves segment of obj_p into but_p
                                   ; should be segment of Button_List


      obj_p = &Current_Buttons[cbutn];
      but_p = &Button_List[obj_p->button_code];
```

Temporary solution:
The temporary solution is to access the variable directly without using pointers.

```
     but_p = &Button_list[Current_Buttons[cbutn].button_code];
```

Signed off 04/29/87 in release 103.30

Number: 5000154245 Product: 8086/8 C 500 64818S001 03.00

Keywords: CODE GENERATOR

One-line description:
Compiler generates MOVSB without init. ES - POINTER -> member = VAR;

Problem:
The following code generates a MOVSB without loading the ES register prior to moving the data. The Source register, SI, uses the DS segment, but the destination register, DI, uses the ES register. In this case the ES register has unknown contents.

```
   "C"
   "8088"
   $FAR_PROC ON$
   $FAR_LIBRARIES ON$
   $SEPARATE_CONST OFF$
```



```
   struct str_1 { short instr1;
                  short instr2;
                  short instr3; };
   struct str_2 { int instr_a;
                  int instr_b;
                  struct str_1 instr_c; };
   extern struct str_1 data_1;

   test()
   {  struct str_2 *jp;
      jp->instr_c = data_1;
          LEA      SI,DS:data_1
          MOV      BX,SS:WORD PTR[BP-2H]
          LEA      DI,DS;[BX+000004H]
          MOV      CX,#+00003H
          CLD
                        <------------Would expect PUSH DS
                                                POP ES  to be here
          REP      MOVSB

           MOVSB uses the ES:DI to calculate destination address,
           but ES has not been loaded


    A similiar example uses a temporary variable:

      same declarations as above
      test()
      { struct str_2 *jp;
        struct str_1 x;
        x = jp->instr_c;
        x = data_1;
      }
```

In this example the ES register gets loaded with the value of thhe SS register. This is correct, but the DS register gets loaded with the value of the ES register. This is incorrect.

Temporary solution:
THe temporary solution is to access the member directly. For example,

```
           test()
              { struct str_1 y;
                y.instr_c = data_1;
              }
```

Signed off 04/29/87 in release 103.30

Number: D200040303 Product: 8086/8 C 500 64818S001 02.01

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner

switch's cases with breaks the compiler generates an infinite loop.
"C"
"processor name"

```
main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.
"C"
"processor name"

```
main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 103.30

Number: D200045559 Product: 8086/8 C 500 64818S001 02.01

One-line description:
File will not compile on the 9000/500.

Temporary solution:
Download the source to the 64000 compile it and then upload the relocatable.

Signed off 04/29/87 in release 103.30

Number: D200059683 Product: 8086/8 C 500 64818S001 03.10

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

"C"
"processor"

```
main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 103.30

Number: D200063396 Product: 8086/8 C 500 64818S001 03.20

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 103.30

Number: D200065987 Product: 8086/8 C 500 64818S001 03.20

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:
"C"
"processor"

```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

"C"
"processor"

```
char  string[]  =  "do it this way";
```

```
main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 103.30

Number: D200066365 Product: 8086/8 C 500 64818S001 03.20

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 103.30

Number: 5000114645 Product: 8086/8 C VAX 64818S003 02.00

One-line description:
Data space cannot exceed 32K.

Signed off 04/29/87 in release 303.50

Number: 5000128959 Product: 8086/8 C VAX 64818S003 02.01

Keywords: CODE GENERATOR

One-line description:
float/double vars. in a subroutine uses MOVESB without init. ES

Problem:
The following example demonstrates that float variables used in subroutines generate incorrect code. This problem also occurs with the use of double variables. The defect is present on all hosts.

```
"C"
"80186"
sub(a)
float a[];
{
float b;
a[1] = b;
}
```

The code assumes that DS=ES in near mode; however, the actual code that is generated never initializes ES to be equilavent to DS before the MOVESB instruction.

Temporary solution:
No known temporary solution.

Signed off 04/29/87 in release 303.50

Number: 5000129817 Product: 8086/8 C VAX 64818S003 03.10

One-line description:
Compiler aborts when incorrectly passing address of array as funct. para

Problem:
The following program gets the expected error (Lvalue expected error) when compiled on the 64100, but it generates a "Comp: c pass1 cannot recover from errors parsing stopped at line xxx" error on the 9000 and VAX.

```
    "C"
    "processor name"
    $PAR_PROC ON$
    $POINTER_SIZE 32$
    extern int Reply();
    extern int CF();
    extern int HD();
    main() {  HD(&Reply,CF());}
```

Customer would like same error message on the 9000/VAX as on the 64100.

Temporary solution:
No known temporary solution.

Signed off 04/29/87 in release 303.50

Number: D200040311 Product: 8086/8 C VAX 64818S003 02.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks, the compiler generates an infinite loop.
"C"
"processor name"

```
main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here, it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.
"C"
"processor name"

```
main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                      case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 303.50

Number: D200059691 Product: 8086/8 C VAX 64818S003 03.10

One-line description:
Compiler is not flagging an undefined structure.

Problem:

The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

"C"
"processor"

```
main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 303.50

Number: D200063404 Product: 8086/8 C VAX 64818S003 03.40

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 303.50

Number: D200065995 Product: 8086/8 C VAX 64818S003 03.40

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:
"C"
"processor"

```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 303.50

Number: D200066373 Product: 8086/8 C VAX 64818S003 03.40

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 303.50

Number: D200030775 Product: 8086/8 PASCAL 64814 02.00

One-line description:
Incorrect code generated for assignment statement.

Problem:
The following program illustrates a code generation problem.

```
PROGRAM test;

   VAR
      a, b, c : BYTE;

   BEGIN
      IF a <> 255 THEN
         b := (c * 10) + a;
   END.
```

The compiler assumes that a register value is valid and does not reload. Since the register value is NOT valid, this produces an error.

Temporary solution:
Use the compiler option $AMNESIA ON$.

Signed off 04/29/87 in release 403.02

Number: D200037325 Product: 8086/8 PASCAL 64814 02.01

One-line description:
Program reboots or aborts with too many errors (64000 / host).

Signed off 04/29/87 in release 403.02

Number: D200055335 Product: 8086/8 PASCAL 64814 03.00

Keywords: CODE GENERATOR

One-line description:
$SEPARATE_CONST OFF$ USED WITH REAL # CONSTS. GENERATES POP CS/PUSH CS.

Problem:
THE FOLLOWING PROGRAM GENERATES TWO ILLEGAL INSTRUCTIONS: POP CS AND PUSH CS.

```
        "80186"

        $POINTER_SIZE 32$
        $SEPARATE_CONST OFF$
        $GLOBPROC ON$

        PROGRAM INIT;

        $GLOBVAR ON$
        VAR CPU : REAL;

        PROCEDURE TEST;
          BEGIN
```

```
          CPU := 9.8304E6;
        END;

      BEGIN
      END.
```

THE PROBLEM ONLY OCCURS WHEN THE $SEPARATE_CONST OFF$ IS USED.

Temporary solution:
DO NOT USE $SEPARATE_CONST OFF$ WITH REAL CONSTANTS.

Signed off 04/29/87 in release 403.02

Number: D200063990 Product: 8086/8 PASCAL 64814 03.01

Keywords: CODE GENERATOR

One-line description:
Record members' addresses are calcul. incorrectly inside the WITH stmnt

Problem:
The address of a record member accessed by a pointer is calculated incorrectly when use in a WITH statment.

```
"80186"
$POINTER_SIZE 32$
PROGRAM PROG_INIT;

TYPE
   INFO = RECORD
          DUMMY1 : INTEGER;
          DUMMY2 : INTEGER;
          END;
   CONTROL = RECORD
             COMMAND : INTEGER;
             NUMBER : INTEGER;
             END;
   ALL_INFOS = RECORD
               SB0 : INFO;
               SBI : ARRAY [1..10] OF CONTROL;
               END;

VAR
   X_AL_INFOS : ^ALL_INFOS;

PROCEDURE PROC_INIT;
   BEGIN
      WITH X_AL_INFOS^.SBI[1] DO
           LES    BX,DS:DWORD PTR DPROG_INIT  (loads addr of record -
                                               type all_infos)
           ADD    BX,#+00008H                 (loads addr of x_al_infos
                                               .sbi[1].command)
        BEGIN
         NUMBER := 50;
           PUSH   #0
           PUSH   #+00032H
```



```
           POP    ES:[BX+0000CH]              (puts value into incorrec
                                               location - should be
                                               BX+00004)
           POP    ES:[BX+0000EH]              (should be BX+00006)
        COMMAND := 20;
           PUSH   #0
           PUSH   #+00014H
           POP    ES:[BX+00008H]              (should be BX)
           POP    ES:[BX+0000AH]              (should be BX+00002)
      END;
   END;
.
```

Temporary solution:
The temporary solution is to not use the WITH statement.
Use the full path name to access the record member.

Signed off 04/29/87 in release 403.02

Number: D200065078 Product: 8086/8 PASCAL 64814 03.01

Keywords: PASS 3 CODE GENERATOR

One-line description:
SHORT JMP generated instead of NEAR JMP when jumping > 32K

Problem:
This problem generates different code on the 64100 than on the 9000 series 500. On the 9000, the code generated is larger than 32K. Whenever it passes 32K, an #1113 error (Program Counters do not agree) is flagged. A NEAR PTR JMP is generated.

On the 64100, the code generated does not cause any errors or warnings, but the jump generated is incorrect. A SHORT JMP has to be made within 32K. It should have been a NEAR PTR JMP.

The code is available on hplsdsb!robin under users/robin/D.hotsite/D.BOR/fmt2_32k.p.

Signed off 04/29/87 in release 403.02

Number: D200050245 Product: 8086/8 PASCAL 300 64814S004 03.00

Keywords: PASS 2

One-line description:
Too many errors, pass2: 80186 (PROCEDURE, WITH statement).

Signed off 04/29/87 in release 403.20

Number: D200051219 Product: 8086/8 PASCAL 300 64814S004 03.00

One-line description:
Incorrect code generated for assignment statement.

Problem:
The following program illustrates a code generation problem.

```
PROGRAM test;

   VAR
      a, b, c : BYTE;

   BEGIN
      IF a <> 255 THEN
         b := (c * 10) + a;
   END.
```

The compiler assumes that a register value is valid and does not reload. Since the register value is NOT valid, this produces an error.

Temporary solution:
Use the compiler option $AMNESIA ON$.

Signed off 04/29/87 in release 403.20

Number: D200051797 Product: 8086/8 PASCAL 300 64814S004 03.00

One-line description:
Program reboots or aborts with too many errors (64000 / host).

Signed off 04/29/87 in release 403.20

Number: D200055517 Product: 8086/8 PASCAL 300 64814S004 03.00

Keywords: CODE GENERATOR

One-line description:
$SEPARATE_CONST OFF$ USED WITH REAL # CONSTS. GENERATES POP CS/PUSH CS.

Problem:
THE FOLLOWING PROGRAM GENERATES TWO ILLEGAL INSTRUCTIONS: POP CS AND PUSH CS.

```
        "80186"

        $POINTER_SIZE 32$
        $SEPARATE_CONST OFF$
```



```
        $GLOBPROC ON$

        PROGRAM INIT;

        $GLOBVAR ON$
        VAR CPU : REAL;

        PROCEDURE TEST;
          BEGIN
            CPU := 9.8304E6;
          END;

        BEGIN
        END.
```

THE PROBLEM ONLY OCCURS WHEN THE $SEPARATE_CONST OFF$ IS USED.

Temporary solution:
DO NOT USE $SEPARATE_CONST OFF$ WITH REAL CONSTANTS.

Signed off 04/29/87 in release 403.20

Number: D200064097 Product: 8086/8 PASCAL 300 64814S004 03.10

Keywords: CODE GENERATOR

One-line description:
Record members' addresses are calcul. incorrectly inside the WITH stmnt

Problem:
The address of a record member accessed by a pointer is calculated incorrectly when use in a WITH statment.

```
"80186"
$POINTER_SIZE 32$
PROGRAM PROG_INIT;

TYPE
   INFO = RECORD
          DUMMY1 : INTEGER;
          DUMMY2 : INTEGER;
          END;
   CONTROL = RECORD
             COMMAND : INTEGER;
             NUMBER : INTEGER;
             END;
   ALL_INFOS = RECORD
               SB0 : INFO;
               SBI : ARRAY [1..10] OF CONTROL;
               END;

VAR
   X_AL_INFOS : ^ALL_INFOS;

PROCEDURE PROC_INIT;
   BEGIN
      WITH X_AL_INFOS^.SBI[1] DO
```

```
            LES    BX,DS:DWORD PTR DPROG_INIT  (loads addr of record -
                                                type all_infos)
            ADD    BX,#+00008H                 (loads addr of x_al_infos
                                                .sbi[1].command)
         BEGIN
          NUMBER := 50;
            PUSH   #0
            PUSH   #+00032H
            POP    ES:[BX+0000CH]              (puts value into incorrec
                                                location - should be
                                                BX+00004)
            POP    ES:[BX+0000EH]              (should be BX+00006)
          COMMAND := 20;
            PUSH   #0
            PUSH   #+00014H
            POP    ES:[BX+00008H]              (should be BX)
            POP    ES:[BX+0000AH]              (should be BX+00002)
         END;
    END;
.
```

Temporary solution:
The temporary solution is to not use the WITH statement.
Use the full path name to access the record member.

Signed off 04/29/87 in release 403.20

Number: D200025908 Product: 8086/8 PASCAL 500 64814S001 01.10

Keywords: PASS 2

One-line description:
Too many errors, pass2: 80186 (PROCEDURE, WITH statement).

Signed off 04/29/87 in release 103.20

Number: D200030783 Product: 8086/8 PASCAL 500 64814S001 02.00

One-line description:
Incorrect code generated for assignment statement.

Problem:
The following program illustrates a code generation problem.

```
PROGRAM test;

   VAR
      a, b, c : BYTE;

   BEGIN
      IF a <> 255 THEN
         b := (c * 10) + a;
   END.
```

The compiler assumes that a register value is valid and does not reload. Since the register value is NOT valid, this produces an error.

Temporary solution:
Use the compiler option $AMNESIA ON$.

Signed off 04/29/87 in release 103.20

Number: D200037333 Product: 8086/8 PASCAL 500 64814S001 02.00

One-line description:
Program reboots or aborts with too many errors (64000 / host).

Signed off 04/29/87 in release 103.20

Number: D200055491 Product: 8086/8 PASCAL 500 64814S001 03.00

Keywords: CODE GENERATOR

One-line description:
$SEPARATE_CONST OFF$ USED WITH REAL # CONSTS. GENERATES POP CS/PUSH CS.

Problem:
THE FOLLOWING PROGRAM GENERATES TWO ILLEGAL INSTRUCTIONS: POP CS
AND PUSH CS.

```
        "80186"

        $POINTER_SIZE 32$
        $SEPARATE_CONST OFF$
```

```
        $GLOBPROC ON$

        PROGRAM INIT;

        $GLOBVAR ON$
        VAR CPU : REAL;

        PROCEDURE TEST;
          BEGIN
            CPU := 9.8304E6;
          END;

        BEGIN
        END.
```

THE PROBLEM ONLY OCCURS WHEN THE $SEPARATE_CONST OFF$ IS USED.

Temporary solution:
DO NOT USE $SEPARATE_CONST OFF$ WITH REAL CONSTANTS.

Signed off 04/29/87 in release 103.20

Number: D200064071 Product: 8086/8 PASCAL 500 64814S001 03.10

Keywords: CODE GENERATOR

One-line description:
Record members' addresses are calcul. incorrectly inside the WITH stmnt

Problem:
The address of a record member accessed by a pointer is calculated incorrectly when use in a WITH statment.

```
"80186"
$POINTER_SIZE 32$
PROGRAM PROG_INIT;

TYPE
   INFO = RECORD
          DUMMY1 : INTEGER;
          DUMMY2 : INTEGER;
          END;
   CONTROL = RECORD
             COMMAND : INTEGER;
             NUMBER : INTEGER;
             END;
   ALL_INFOS = RECORD
               SB0 : INFO;
               SBI : ARRAY [1..10] OF CONTROL;
               END;

VAR
   X_AL_INFOS : ^ALL_INFOS;

PROCEDURE PROC_INIT;
   BEGIN
      WITH X_AL_INFOS^.SBI[1] DO
```



```
            LES    BX,DS:DWORD PTR DPROG_INIT  (loads addr of record -
                                                type all_infos)
            ADD    BX,#+00008H                 (loads addr of x_al_infos
                                                .sbi[1].command)
         BEGIN
          NUMBER := 50;
             PUSH   #0
             PUSH   #+00032H
             POP    ES:[BX+0000CH]              (puts value into incorrec
                                                location - should be
                                                BX+00004)
             POP    ES:[BX+0000EH]              (should be BX+00006)
          COMMAND := 20;
             PUSH   #0
             PUSH   #+00014H
             POP    ES:[BX+00008H]              (should be BX)
             POP    ES:[BX+0000AH]              (should be BX+00002)
         END;
    END;
.
```

Temporary solution:
The temporary solution is to not use the WITH statement.
Use the full path name to access the record member.

Signed off 04/29/87 in release 103.20

Number: D200025916 Product: 8086/8 PASCAL VAX 64814S003 01.10

Keywords: PASS 2

One-line description:
Too many errors, pass 2: 80186 (PROCEDURE, WITH statement).

Signed off 04/29/87 in release 303.30

Number: D200030791 Product: 8086/8 PASCAL VAX 64814S003 02.00

One-line description:
Incorrect code generated for assignment statement.

Problem:
The following program illustrates a code generation problem.

```
PROGRAM test;

   VAR
      a, b, c : BYTE;

   BEGIN
      IF a <> 255 THEN
         b := (c * 10) + a;
   END.
```

The compiler assumes that a register value is valid and does not reload. Since the register value is NOT valid, this produces an error.

Temporary solution:
Use the compiler option $AMNESIA ON$.

Signed off 04/29/87 in release 303.30

Number: D200037341 Product: 8086/8 PASCAL VAX 64814S003 02.00

One-line description:
Program reboots or aborts with too many errors (64000 / host).

Signed off 04/29/87 in release 303.30

Number: D200055509 Product: 8086/8 PASCAL VAX 64814S003 03.00

Keywords: CODE GENERATOR

One-line description:
$SEPARATE_CONST OFF$ USED WITH REAL # CONSTS. GENERATES POP CS/PUSH CS.

Problem:
THE FOLLOWING PROGRAM GENERATES TWO ILLEGAL INSTRUCTIONS: POP CS AND PUSH CS.

```
        "80186"

        $POINTER_SIZE 32$
        $SEPARATE_CONST OFF$
```



```
        $GLOBPROC ON$

        PROGRAM INIT;

        $GLOBVAR ON$
        VAR CPU : REAL;

        PROCEDURE TEST;
          BEGIN
            CPU := 9.8304E6;
          END;

        BEGIN
        END.
```

THE PROBLEM ONLY OCCURS WHEN THE $SEPARATE_CONST OFF$ IS USED.

Temporary solution:
DO NOT USE $SEPARATE_CONST OFF$ WITH REAL CONSTANTS.

Signed off 04/29/87 in release 303.30

Number: D200064089 Product: 8086/8 PASCAL VAX 64814S003 03.20

Keywords: CODE GENERATOR

One-line description:
Record members' addresses are calcul. incorrectly inside the WITH stmnt

Problem:
The address of a record member accessed by a pointer is calculated incorrectly when use in a WITH statment.

```
"80186"
$POINTER_SIZE 32$
PROGRAM PROG_INIT;

TYPE
   INFO = RECORD
          DUMMY1 : INTEGER;
          DUMMY2 : INTEGER;
          END;
   CONTROL = RECORD
             COMMAND : INTEGER;
             NUMBER : INTEGER;
             END;
   ALL_INFOS = RECORD
               SB0 : INFO;
               SBI : ARRAY [1..10] OF CONTROL;
               END;

VAR
   X_AL_INFOS : ^ALL_INFOS;

PROCEDURE PROC_INIT;
   BEGIN
      WITH X_AL_INFOS^.SBI[1] DO
```

```
            LES    BX,DS:DWORD PTR DPROG_INIT  (loads addr of record -
                                                type all_infos)
            ADD    BX,#+00008H                 (loads addr of x_al_infos
                                                .sbi[1].command)
        BEGIN
         NUMBER := 50;
            PUSH   #0
            PUSH   #+00032H
            POP    ES:[BX+0000CH]              (puts value into incorrec
                                                location - should be
                                                BX+00004)
            POP    ES:[BX+0000EH]              (should be BX+00006)
         COMMAND := 20;
            PUSH   #0
            PUSH   #+00014H
            POP    ES:[BX+00008H]              (should be BX)
            POP    ES:[BX+0000AH]              (should be BX+00002)
        END;
    END;
.
```

Temporary solution:
The temporary solution is to not use the WITH statement.
Use the full path name to access the record member.

Signed off 04/29/87 in release 303.30

Number: D200021790 Product: HOST SOFTWARE / VAX 64882 01.10

One-line description:
File name conversion (transfer) is inconsistent with COMP and ASM.

Signed off 04/29/87 in release 202.00

Number: D200045088 Product: HOST SOFTWARE / VAX 64882 01.20

Keywords: TRANSFER

One-line description:
Insufficient examples in the HELP entry.

Signed off 04/29/87 in release 202.00

Number: D200046102 Product: HOST SOFTWARE / VAX 64882 01.20

One-line description:
Transfer may not function across VAX-cluster.

Signed off 04/29/87 in release 202.00

Number: D200047951 Product: HOST SOFTWARE / VAX 64882 01.20

Keywords: HIGH SPEED LINK

One-line description:
Initializing the HSL may require more than one shift/reset on the 64000.

Problem:
After CSIB is run, Mapbus (SYSTEM_1) is spawned, and the 64000 master is reset to allow the Mapbus to complete, it appears that the Mapbus has successfully completed. But, subsequently manually running a Mapbus does not work. Furthermore, when the HSL is in this state, transfers will not complete too.

Temporary solution:
It might be necessary to do two shift/resets on the 64000 master.

Signed off 04/29/87 in release 202.00

Number: D200048017 Product: HOST SOFTWARE / VAX 64882 01.20

Keywords: HIGH SPEED LINK

One-line description:
HSLSTOP doesn't work if MAPBUS is pending.

Problem:
If MAPBUS(SYSTEM_1) is pending for CSIBn, HSLSTOP/HSL=n will not stop the CSIBn process.

Temporary solution:
Use "$STOP PROCESS/ID= "

Signed off 04/29/87 in release 202.00

Number: D200048140 Product: HOST SOFTWARE / VAX 64882 01.50

Keywords: TRANSFER

One-line description:
CLUSTER-CLUSTER transfers don't work.

Problem:
Cluster-Cluster transfers don't work and may crash the VAX.

Temporary solution:
Transfer from a cluster to a temporary file on the VAX, then transfer the temporary file to the second cluster.

Signed off 04/29/87 in release 202.00

Number: D200054775 Product: HOST SOFTWARE / VAX 64882 01.60

Keywords: RCMAIN

One-line description:
RCMAIN/VERBOSE not described in the HELP file.

Signed off 04/29/87 in release 202.00

Number: D200065680 Product: HOST SOFTWARE / VAX 64882 01.70

One-line description:
Misspellings in HPINSTALL.COM can cause %F-ERROR.

Problem:
Line # 272 has HPI$PROTDUCTS instead of HPI$PRODUCTS.

Line # 281 has HPI_End_copy_Product instead of HPI_End_copy_Products
(page 394)

Line # 421 has an inconsequential misspelling of represesetative.

Signed off 04/29/87 in release 202.00

Number: D200067512 Product: HOST SOFTWARE / VAX 64882 01.70

One-line description:
HSL will not start with most 64000 printers (introduced in 1.7)

Signed off 04/29/87 in release 202.00

Number: D200047845 Product: HOST SOFTWARE / VAX 64882 01.20

Keywords: TRANSFER

One-line description:
TRANSFER does not timeout.

Signed off 04/29/87 in release 202.00

Number: D200048041 Product: HOST SOFTWARE / VAX 64882 01.20

Keywords: HIGH SPEED LINK

One-line description:
IBDRIVER conficts with existing driver on the system.

Signed off 04/29/87 in release 202.00

Number: D200055012 Product: HOST SOFTWARE / VAX 64882 01.60

Keywords: MAPBUS

One-line description:
Define MAPBUS as a verb in HPTABLES.CLD instead of a symbol in HPSETUP.

Signed off 04/29/87 in release 202.00

Number: D200060285 Product: NSC800 EMULATION 64292 01.02

One-line description:
Incorrect Inverse Assembly with State when restart active

Problem:
The inverse assembler for State gives incorrect IA when restart is active.

Example:

```
0019     LD   A,**
0078       xx refresh (restart req)
001A       01 memory read (restart req)
```

Should be:

```
0019     LD   A,01
```

Signed off 04/29/87 in release 201.03

Number: D200067470 Product: NSC800 EMULATION 64292 01.02

One-line description:
NSC800 cannot access the last 256 byte block of user memory.

Problem:
It is not possible to access the last 256 block of user memory under the following conditions:

1) running with a slow external oscillator, freq < 2MHz

2) the very last entry in the memory map is user memory.

3) you are using revision 1.02 of the NSC800 Emulation software.

When accessing the last block, typically you will always read zeros.

Temporary solution:
Two work-arounds exist for this problem.

1) Add a dummy entry to the memory map following the last block of user memory that was mapped previously. The dummy map entry should be the last entry in numerical order.
2) Modify the memory map so that the last entry is emulation memory rather than user memory, since this problem only appears if the last entry in the memory map is user memory.

If for example, you have memory mapped I/O located at 0FF00H - 0FFFFH, then the two work-arounds mentioned above will be of no help to you. In this special case, the best work-around is to operate with an older revision of NSC800 Emulation software, such as revision 1.01

Signed off 04/29/87 in release 201.03

Number: D200067488 Product: NSC800 EMULATION 64292 01.02

One-line description:
"modify register PC" immediately after "load <absolute_file>" fails

Problem:
"modify register PC" immediately after loading an absolute file will fail to modify the PC.

Temporary solution:
Since this problem only occurs when the "modify register PC" command is issued immediately after the "load" command, the best work-around is to use the "reset" command after the load, or simply repeat the modify register command until it does work.

Signed off 04/29/87 in release 201.03

Number: D200072199 Product: OPERATING SYSTEM 64100 02.06

One-line description:
MAIN Assemb stops table interpretation for expressions delinited by "."

Problem:
The 68000 assembler was demostrating problem with statements of the form

```
  mov 0[A0,D0.L],2[A0,D0.L]
```

terminating on the first ".L" in the expression handler of the 64000 table interpreter. The table trace showed that table interpretation had ceased in the EXPRESSION pseudoinstruction of the table.

Dave Ritchie

Signed off 04/29/87 in release 002.07

Number: D200061614 Product: USER DEF ASSEMB 300 64851S004 01.00

One-line description:
Problem with timemark in hosted assemblers.

Problem:
The Hosted Assembler produces a timemark in the relocatable file which has an offset from the actual time i.e. minus 14 hours.

Signed off 04/29/87 in release 401.20

Number: D200062653 Product: USER DEF ASSEMB 300 64851S004 01.00

One-line description:
EQU pseudo with 0LLH for an operand may halt assembly.

Problem:
When an EQU statement is followed by 0LLH as an operand, the assembly process is halted. This should not happen. An error should be flagged to the affect of an invalid operand, but the assembly process should not halt. Assemble this file with listing on, and notice that the statements after the EQU 0LLH statement are assembled. Then assemble test009 (attached to this document under the file name EQU) with listing on, and notice that there are only 107 lines listed (terminating with EQU 0LLH) and there should be 271 lines listed.

```
 EQU 0LLH
 STA 41H
 LDA 41H
 END
```

Signed off 04/29/87 in release 401.20

Number: D200063248 Product: USER DEF ASSEMB 300 64851S004 01.10

Keywords: LINKER

One-line description:
Linker does not correctly handle "NO LOAD" files.

Signed off 04/29/87 in release 401.20

Number: D200067454 Product: USER DEF ASSEMB 300 64851S004 01.10

One-line description:
Assembler aborts when full path name is specified.

Signed off 04/29/87 in release 401.20

Number: D200065011 Product: USER DEF ASSEMB 300 64851S004 01.10

One-line description:
Assembler trys to assemble .A files.

Problem:
If you mistakenly try to assembler a .A file the assembler does not report an error and it hangs.

Signed off 04/29/87 in release 401.20

Number: 5000149211 Product: USER DEF ASSEMB 500 64851S001 01.40

One-line description:
Comma at the end of a HEX pseudo statement causes the assembler to hang.

Problem:
If a comma is incorrectly placed after a pseudo HEX instruction
the assembler hangs.

"processor"

```
                HEX     00,
```

This is true only on the host machines.

Signed off 04/29/87 in release 101.60

Number: D200055384 Product: USER DEF ASSEMB 500 64851S001 01.30

One-line description:
ASM is unable to assemble a file accessed across lan via a netunam.

Problem:
asm on the series 500 when accessing a file across LAN using
netunam will not assemble the file. Example:

```
   $netunam /net/remote_machine uid:
   Password:
   $cd /net/remote_machine/some_directory
   $asm file.s
 asm: Termination, Input file not found.(line 0)
```

Signed off 04/29/87 in release 101.60

Number: D200061598 Product: USER DEF ASSEMB 500 64851S001 01.40

One-line description:
Problem with timemark in hosted assemblers.

Problem:
The Hosted Assembler produces a timemark in the relocatable file
which has an offset from the actual time i.e. minus 14 hours.

Signed off 04/29/87 in release 101.60

Number: D200063230 Product: USER DEF ASSEMB 500 64851S001 01.50

Keywords: LINKER

One-line description:
Linker does not correctly handle "NO LOAD" files.

Problem:

Temporary solution:

Signed off 04/29/87 in release 101.60

Number: D200067439 Product: USER DEF ASSEMB 500 64851S001 01.50

One-line description:
Assembler aborts when full path name is specified.

Problem:
Specifying the full path name of a file when invoking the assembler causes the following error to be flagged.

asm: Can not recover from errors on line 1. (11)

Temporary solution:
Copy the file you wish to assemble to your current directory.

Signed off 04/29/87 in release 101.60

Number: D200064840 Product: USER DEF ASSEMB 500 64851S001 01.50

One-line description:
Assembler trys to assemble .A files.

Problem:
If you mistakenly try to assembler a .A file the assembler does not report an error and it hangs.

Signed off 04/29/87 in release 101.60

Number: 5000143370 Product: USER DEF ASSEMB VAX 64851S003 01.04

Keywords: LINKER

One-line description:
Linker does not correctly handle "NO LOAD" files.

Signed off 04/29/87 in release 301.60

Number: D200060830 Product: USER DEF ASSEMB VAX 64851S003 01.50

Keywords: LINKER

One-line description:
Displacement > 32K error being flagged when it should not be.

Problem:
The problem is that their link contains a link_sym (.L) file. It appears that when a link is done with a link_sym file and the libraries are load at a location greater than 8000H this error (Displacement > 32K) is flagged. The errors are flagged at BSR instrucions which are well within 32K.

Signed off 04/29/87 in release 301.60

Number: D200062646 Product: USER DEF ASSEMB VAX 64851S003 01.50

One-line description:
EQU pseudo with 0LLH for an operand may halt assembly.

Problem:
When an EQU statement is followed by 0LLH as an operand, the assembly process is halted. This should not happen. An error should be flagged to the affect of an invalid operand, but the assembly process should not halt. Assemble this file with listing on, and notice that the statements after the EQU 0LLH statement are assembled. Then assemble test009 (attached to this document under the file name EQU) with listing on, and notice that there are only 107 lines listed (terminating with EQU 0LLH) and there should be 271 lines listed.

```
 EQU 0LLH
 STA 41H
 LDA 41H
 END
```

Signed off 04/29/87 in release 301.60

Number: D200065003 Product: USER DEF ASSEMB VAX 64851S003 01.50

One-line description:
Assembler trys to assemble .A files.

Problem:
If you mistakenly try to assembler a .A file the assembler does not report an error and it hangs.

Signed off 04/29/87 in release 301.60

Number: 2700003921 Product: Z80/NSC800 C 64824 00.00

One-line description:
Changes to pointers to unions does not work properly in C language.

Problem:
The compiler does not recognize changes in pointers to unions. The original value will be used as the new data if the compiler thinks it is still available. AMNESIA has no effect.

Signed off 04/29/87 in release 401.04

Number: 2700003939 Product: Z80/NSC800 C 64824 00.00

One-line description:
Certain single argument Rvalues will not compile correctly.

Problem:
Single argument Rvalues which are operated on by the:
++, --, +=, -=, etc. operators, and assigned to an indirect structure member of an indirectly accesed structure will not compile correctly in certain cases.

Signed off 04/29/87 in release 401.04

Number: 2700004093 Product: Z80/NSC800 C 64824 00.00

One-line description:
Library routine REAL_SUB modifies DE register pair.

Problem:
LIBRARY ROUTINE REAL_SUB MODIFIES DE REGISTER PAIR

Signed off 04/29/87 in release 401.04

Number: 2700005603 Product: Z80/NSC800 C 64824 01.01

Keywords: CODE GENERATOR

One-line description:
Registers used by Zbshift loaded incorrectly after structure reference.

Temporary solution:
MASK THE BITS BEFORE THE SHIFT IS EXECUTED. SEE EXAMPLE BELOW.

```
"C"
"Z80"
typedef struct {
        char hh, mm, ss;}
        time;

int
conv_time(tm_ptr)
time *tm_ptr;
{extern int tab_sH[], tab_mL[], tabmH[], tab_hL[], tab_hH[];
static int acc;
acc += tab_sH[(tm_ptr-> ss &0x50) >>4];
```

```
/*Use above line rather than this next line*/
acc += tab_sH[(tm_ptr-> ss >>4) & 0x05];
}
```

Signed off 04/29/87 in release 401.04

Number: 5000139204 Product: Z80/NSC800 C 64824 01.03

Keywords: CODE GENERATOR

One-line description:
Character isn't converted to int before calculations use it

Problem:
Kernigan and Ritchie states that a character is converted to an integer before calculations use the char variable. Our compiler does not conver t the character to an integer prior to any calculations.

For example:

```
   "C"
   "8086"
   main() {
     char c;
     int i;
     i = ((c<< 4) *5)/i;

 AX register if c = 0FFH
 ------
  xxxx     MOV  CL,#+00004H   {moves 4 into counter}
  00xx     MOV  AH,#0         {00h into AH}
  00FF     MOV  AL,SS:8YTE PTR[8P-00003H]  {loads c into AL}
  00F0     SHL  AL,CL       {shifts left 4 c ;however, it loses the uppe
                              r byte because it was not SHL AX,CL}
   }
```

The character is not being treated as an integer. Making this SHL AX,CL would fix the problem.

Emulating the generated code confirmed that the high byte (4 places) was not being shifted into AH.

Temporary solution:
Type cast c to be an integer before using it in the expression.

Signed off 04/29/87 in release 401.04

Number: D200011148 Product: Z80/NSC800 C 64824 01.00

Keywords: PASS 1

One-line description:
Functions invoded via function pointers may JSR the wrong location.

Problem:
When the typedef statement is used to define pointers to functions, and this pointer type is used in a cast of a variable array to invoke

code stored in that array, program execution may transfer to the wrong location. For example, in the following code the simple call to code_array fails while the call and assignment to p works correctly:

```
typedef int(*PFI)();  /* PFI a pointer to int functions */
int code_array[100];  /* array contains code */
PFI p;                /* p a pointer of type PFI */

pfibug()
{
  (*((PFI) code_array))();  /* fails in JSR to code_array */
  (*(p=(PFI)code_array))(); /* assignment and JSR successful */
}
```

Temporary solution:
Set up a dummy variable and perform an assignment to it when doing this type of operation.

Signed off 04/29/87 in release 401.04

Number: D200011221 Product: Z80/NSC800 C 64824 01.00

Keywords: PASS 1

One-line description:
Unsigned integers treated as signed when subtracted from pointers.

Problem:
When an unsigned short or integer is used as an offset to a pointer, the unsigned will be treated as a signed when doing pointer calculations. Offsets large enough to set the sign bit will be interpreted as a negative offset when the offset is subtracted from a pointer. The following code exibits the problem if offset is greater than 32767 dec.

```
unsigned offset;
struct { int a,b,c;
       } *ptr;
unsigned long x;

main ()
{
  x = ptr - offset;  /* The compiler will generate code negating   */
}                    /* offset for the "-" operation.              */
```

Temporary solution:
Cast the offset in the expression as the next larger integer.
ie. x = ptr - (unsigned long)offset;

Signed off 04/29/87 in release 401.04

Number: D200013300 Product: Z80/NSC800 C 64824 01.00

Keywords: CODE GENERATOR

One-line description:
Assigning a ptr. after its post incr/decr. gives incorrect value.

Problem:

Pointer assignment after a post increment or decrement to that pointer stores incorrect value. The following is an illustration:

```
"C"
"PROCESSOR_NAME"

unsigned short fct(g)
unsigned short *g;
{
   unsigned short a,b;
   b=*g;
   *g++;
   a=*g;
}
```

The first assignment statement stores the contents of what g is point-to in the accumulator. Once the pointer is incremented, the compiler loads the accumulator (which still has the previous value) into the variable a. The compiler is false remembering the value in the accumulator as the current contents of what g is pointing to.

Temporary solution:
Turn $AMNESIA ON$ to force the reload of the accumulator from the BC register pair.

Signed off 04/29/87 in release 401.04

Number: D200015966 Product: Z80/NSC800 C 64824 01.01

Keywords: PASS 2

One-line description:
Pass 2 error #1006 in if construct when subtracting a const. from a var.

Problem:
The following code generates a Pass 2 error #1006.

```
"C"
"Z80"
#define NULL 0
fct(parm)
int parm;
{ if ( parm - NULL )
       parm = 10;
}
```

If "parm" is defined as an integer pointer, two loads are performed but no other code is generated to check for zero value.

Temporary solution:
Check for value of parm by using the "==" conditional operator.

Signed off 04/29/87 in release 401.04

Number: D200022301 Product: Z80/NSC800 C 64824 01.01

Keywords: CODE GENERATOR

One-line description:
Operating on parm. in function call generates incorrect code.

Temporary solution:
Func_B(retain1)
The temporary fix is to pass a varible that is updated before the procedure call.

```
{int retain1
 int Func_B();

 if (parm_a != 0)
 {retain1 = !retain1
```

Signed off 04/29/87 in release 401.04

Number: D200022624 Product: Z80/NSC800 C 64824 01.01

Keywords: CODE GENERATOR

One-line description:
Pointer addressing wrong location after it has been updated.

Problem:
Using a pointer immediately after it has been updated results in an improper memory location being addressed. The block of code below demostrates this.

```
"C"
"Z80"
{char
   *a,b;
   a= 0xC082;
   a= *a;
   b= *a;  /*"b" is getting what "a" used to point.  "a" is not being
             updated*/
```

Temporary solution:
Use $AMNESIA ON$ option directly after pointer is updated.

```
{char *a,b;
 a= 0xc082;
 a= *a;
 $AMNESIA ON$
 b= *a;
}
```

Signed off 04/29/87 in release 401.04

Number: D200034918 Product: Z80/NSC800 C 64824 01.01

One-line description:
Incorredt or NO listing file produced if fatal pass 2 errors (#10xx)

Problem:

```
C
Z80
#define NULL 0
ct(parm)
int parm;
if (parm - NULL)
    parm=10;
}
```

Signed off 04/29/87 in release 401.04

Number: D200037697 Product: Z80/NSC800 C 64824 01.01

One-line description:
DIF AND WRONG CODE PRODUCED IF ARRAY ELEMENT ASSIGNED RESULT OF INDIRECT

Problem:
Situations where an array element is assigned the result of an array taken indirect may produce code on the 9000 that is different from th 64000 and the VAX, due to an uneeded reload of HL, however, in both c generated code is wrong because the HL register is overwritten by the preceeding calculations.

Sample code:

```
 C
Z80

typedef int = PTR_TYPE;
long *dest_array[2];
int index;
struct STRUCT_TYPE  /* may be union or struct */
{ long filler;  /* a pointer & one or more other items */
 int *i_ptr;  /* requiring 4 or more bytes in any order *
 }

func_1(param)
struct STRUCT_TYPE param; /* parameter must be of STRUCT_TYPE */
 {
 struce STRUCT_TYPE local_strucr[2];  /* must be local array of type
         /* STRUCT_TYPE - size >= 2 */

/* line generating incorrect code */
 dest_array[index] = (* ((PTR_TYPE*) 0x0f1801) [33]);
   /* cast any type of ptr - requires typedef */
   /* constant must be long */
 }
```

Signed off 04/29/87 in release 401.04

Number: D200040410 Product: Z80/NSC800 C 64824 01.01

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.


```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 401.04

Number: D200059865 Product: Z80/NSC800 C 64824 01.02

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 401.04

Number: D200063032 Product: Z80/NSC800 C 64824 01.03

One-line description:
Funct calls via pointers with parms cause subsequent stack ref errors

Problem:
When functions are called via pointers and are passed parameters, subsequent references to stack relative objects will be incorrect. The following code is an example of this problem:

```
"C"
"processor name"
extern int called_func();
typedef int (*PFI)();
PFI call_ptr = called_func;

main()
{
  int local;

  local = 6;              (*variable is accessed correctly*)
  (*(call_ptr() (1,2);    (*function call via pointer with parameters*)
  local = 3;              (*wrong location accessed*)
}
```

Signed off 04/29/87 in release 401.04

Number: D200063578 Product: Z80/NSC800 C 64824 01.03

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 401.04

Number: D200066167 Product: Z80/NSC800 C 64824 01.03

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";
```



```
main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 401.04

Number: D200051961 Product: Z80/NSC800 C 300 64824S004 01.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"processor name"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"processor name"

main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                    case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 401.20

Number: D200059899 Product: Z80/NSC800 C 300 64824S004 01.00

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"
```



```
main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 401.20

Number: D200063289 Product: Z80/NSC800 C 300 64824S004 01.10

Keywords: CODE GENERATOR

One-line description:
Character isn't converted to int before calculations use it

Problem:
Kernigan and Ritchie states that a character is converted to an integer before calculations use the char variable. Our compiler does not conver t the character to an integer prior to any calculations.

For example:

```
   "C"
   "8086"
   main() {
     char c;
     int i;
     i = ((c<< 4) *5)/i;

 AX register if c = 0FFH
 ------
  xxxx     MOV  CL,#+00004H   {moves 4 into counter}
  00xx     MOV  AH,#0         {00h into AH}
  00FF     MOV  AL,SS:BYTE PTR[BP-00003H]  {loads c into AL}
  00F0     SHL  AL,CL       {shifts left 4 c ;however, it loses the uppe
                              r byte because it was not SHL AX,CL}
   }
```

The character is not being treated as an integer. Making this SHL AX,CL would fix the problem.

Emulating the generated code confirmed that the high byte (4 places) was not being shifted into AH.

Temporary solution:
Type cast c to be an integer before using it in the expression.

Signed off 04/29/87 in release 401.20

Number: D200063602 Product: Z80/NSC800 C 300 64824S004 01.10

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 401.20

Number: D200064873 Product: Z80/NSC800 C 300 64824S004 01.10

One-line description:
Funct calls via pointers with parms cause subsequent stack ref errors

Problem:
When functions are called via pointers and are passed parameters, subsequent references to stack relative objects will be incorrect. The following code is an example of this problem:

```
"C"
"processor name"
extern int called_func();
typedef int (*PFI)();
PFI call_ptr = called_func;

main()
{
  int local;

  local = 6;              (*variable is accessed correctly*)
  (*(call_ptr() (1,2);    (*function call via pointer with parameters*)
  local = 3;              (*wrong location accessed*)
}
```

Signed off 04/29/87 in release 401.20

Number: D200066191 Product: Z80/NSC800 C 300 64824S004 01.10

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
```



```
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 401.20

Number: D200066522 Product: Z80/NSC800 C 300 64824S004 01.10

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 401.20

Number: D200015982 Product: Z80/NSC800 C 500 64824S001 01.00

Keywords: PASS 2

One-line description:
Pass 2 Error #1006 when subracting a const. from a var. in an if constr.

Problem:
The following code generates a Pass 2 error #1006.

```
"C"
"Z80"
#define NULL 0
fct(parm)
int parm;
{ if ( parm - NULL )
       parm = 10;
}
```

If "parm" is defined as an integer pointer, two loads are performed but no other code is generated to check for zero value.

Temporary solution:
Check for value of parm by using the "==" conditional operator.

Signed off 04/29/87 in release 101.60

Number: D200025726 Product: Z80/NSC800 C 500 64824S001 01.10

Keywords: CODE GENERATOR

One-line description:
Assigning a ptr. after its post incr/decr. gives incorrect value.

Problem:
Pointer assignment after a post increment or decrement to that pointer stores incorrect value. The following is an illustration:

```
"C"
"PROCESSOR_NAME"

unsigned short fct(g)
unsigned short *g;
{
   unsigned short a,b;
   b=*g;
   *g++;
   a=*g;
}
```

The first assignment statement stores the contents of what g is point-to in the accumulator. Once the pointer is incremented, the compiler loads the accumulator (which still has the previous value) into the variable a. The compiler is false remembering the value in the accumulator as the current contents of what g is pointing to.

Temporary solution:
Turn $AMNESIA ON$ to force the reload of the accumulator from the BC register pair.

Signed off 04/29/87 in release 101.60

Number: D200040428 Product: Z80/NSC800 C 500 64824S001 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
       int  c;
            switch(c){
                 case1:         break;
                 default:       switch(c){
                                   case 2: break;
                                }
                                break;
            }
}
```

Signed off 04/29/87 in release 101.60

Number: D200059873 Product: Z80/NSC800 C 500 64824S001 01.40

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
```

```
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 101.60

Number: D200063263 Product: Z80/NSC800 C 500 64824S001 01.50

Keywords: CODE GENERATOR

One-line description:
Character isn't converted to int before calculations use it

Problem:
Kernigan and Ritchie states that a character is converted to an integer before calculations use the char variable. Our compiler does not conver t the character to an integer prior to any calculations.

For example:

```
   "C"
   "8086"
   main() (
     char c;
     int i;
     i = ((c<< 4) *5)/i;

 AX register if c = 0FFH
 ------
  xxxx     MOV  CL,#+00004H   {moves 4 into counter}
  00xx     MOV  AH,#0         {00h into AH)
  00FF     MOV  AL,SS:BYTE PTR[BP-00003H]  {loads c into AL}
  00F0     SHL  AL,CL       {shifts left 4 c ;however, it loses the uppe
                             r byte because it was not SHL AX,CL}
   }
```

The character is not being treated as an integer. Making this SHL AX,CL would fix the problem.

Emulating the generated code confirmed that the high byte (4 places) was not being shifted into AH.

Temporary solution:
Type cast c to be an integer before using it in the expression.

Signed off 04/29/87 in release 101.60

Number: D200063586 Product: Z80/NSC800 C 500 64824S001 01.50

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 101.60

Number: D200064857 Product: Z80/NSC800 C 500 64824S001 01.50

One-line description:
Funct calls via pointers with parms cause subsequent stack ref errors

Problem:
When functions are called via pointers and are passed parameters, subsequent references to stack relative objects will be incorrect. The following code is an example of this problem:

```
"C"
"processor name"
extern int called_func();
typedef int (*PFI)();
PFI call_ptr = called_func;

main()
{
  int local;

  local = 6;              (*variable is accessed correctly*)
  (*(call_ptr() (1,2);    (*function call via pointer with parameters*)
  local = 3;              (*wrong location accessed*)
}
```

Signed off 04/29/87 in release 101.60

Number: D200066175 Product: Z80/NSC800 C 500 64824S001 01.50

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
```

```
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 101.60

Number: D200066506 Product: Z80/NSC800 C 500 64824S001 01.50

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 101.60

Number: D200015974 Product: Z80/NSC800 C VAX 64824S003 01.00

Keywords: PASS 2

One-line description:
Pass 2 Error #1006 when subtracting a const. from a var. in an if constr

Problem:
The following code generates a Pass 2 error #1006.

```
"C"
"Z80"
#define NULL 0
fct(parm)
int parm;
{ if ( parm - NULL )
       parm = 10;
}
```

If "parm" is defined as an integer pointer, two loads are performed but no other code is generated to check for zero value.

Temporary solution:
Check for value of parm by using the "==" conditional operator.

Signed off 04/29/87 in release 301.90

Number: D200025734 Product: Z80/NSC800 C VAX 64824S003 01.10

Keywords: CODE GENERATOR

One-line description:
Assigning a ptr. after its post incr/decr. gives incorrect value.

Problem:
Pointer assignment after a post increment or decrement to that pointer stores incorrect value. The following is an illustration:

```
"C"
"PROCESSOR_NAME"

unsigned short fct(g)
unsigned short *g;
{
   unsigned short a,b;
   b=*g;
   *g++;
   a=*g;
}
```

The first assignment statement stores the contents of what g is point-to in the accumulator. Once the pointer is incremented, the compiler loads the accumulator (which still has the previous value) into the variable a. The compiler is false remembering the value in the accumulator as the current contents of what g is pointing to.

Temporary solution:
Turn $AMNESIA ON$ to force the reload of the accumulator from the BC register pair.

Signed off 04/29/87 in release 301.90

Number: D200040436 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 301.90

Number: D200059881 Product: Z80/NSC800 C VAX 64824S003 01.50

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
```



```
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 301.90

Number: D200063271 Product: Z80/NSC800 C VAX 64824S003 01.80

Keywords: CODE GENERATOR

One-line description:
Character isn't converted to int before calculations use it

Problem:
Kernigan and Ritchie states that a character is converted to an integer before calculations use the char variable. Our compiler does not convert the character to an integer prior to any calculations.

For example:

```
   "C"
   "8086"
   main() {
     char c;
     int i;
     i = ((c<< 4) *5)/i;

 AX register if c = 0FFH
 ------
  xxxx     MOV  CL,#+00004H   {moves 4 into counter}
  00xx     MOV  AH,#0         {00h into AH}
  00FF     MOV  AL,SS:BYTE PTR[BP-00003H]  {loads c into AL}
  00F0     SHL  AL,CL       {shifts left 4 c ;however, it loses the uppe
                             r byte because it was not SHL AX,CL}
   }
```

The character is not being treated as an integer. Making this SHL AX,CL would fix the problem.

Emulating the generated code confirmed that the high byte (4 places) was not being shifted into AH.

Temporary solution:
Type cast c to be an integer before using it in the expression.

Signed off 04/29/87 in release 301.90

Number: D200063594 Product: Z80/NSC800 C VAX 64824S003 01.80

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 301.90

Number: D200064865 Product: Z80/NSC800 C VAX 64824S003 01.80

One-line description:
Funct calls via pointers with parms cause subsequent stack ref errors

Problem:
When functions are called via pointers and are passed parameters, subsequent references to stack relative objects will be incorrect. The following code is an example of this problem:

```
"C"
"processor name"
extern int called_func();
typedef int (*PFI)();
PFI call_ptr = called_func;

main()
{
  int local;

  local = 6;              (*variable is accessed correctly*)
  (*(call_ptr() (1,2);    (*function call via pointer with parameters*)
  local = 3;              (*wrong location accessed*)
}
```

Signed off 04/29/87 in release 301.90

Number: D200066183 Product: Z80/NSC800 C VAX 64824S003 01.80

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
```



```
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 301.90

Number: D200066514 Product: Z80/NSC800 C VAX 64824S003 01.80

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 301.90

Number: 5000099176 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: IF

One-line description:
IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}

Problem:

```
VAR B1, B2 : BYTE;

BEGIN
IF B1 (>|<|=|<=|>=) B2 THEN
B1 := B1 - 1;   {THE REGISTER CONTAINING B1 IS DECREMENTED, THEN
                 OVERWRITTEN BEFORE IT IS SAVED IN MEMORY}
```

Temporary solution:

```
$AMNESIA +$
```

Signed off 04/29/87 in release 301.04

Number: 5000105841 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for adding one char to another.

Problem:

```
VAR
SRC, DEST : CHAR;

BEGIN
DEST := DEST + SRC;  {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 04/29/87 in release 301.04

Number: 5000146407 Product: Z80/NSC800PASCAL 64823 01.02

One-line description:
Error #1006 when accessing an element of a two-dimensional array.

Problem:
The following code generates Error #1006:

```
"processor name"
PROGRAM ESSAI;
TYPE
  STRING_20=PACKED ARRAY[0..20] OF CHAR;
  TAB_1=ARRAY[1..10] OF STRING_20;
  TAB=ARRAY[1..2] OF TAB_1;
VAR
  V:TAB;
BEGIN
  V[1,1]:="A";       (*Error 1006*)
```



```
END.
```

Signed off 04/29/87 in release 301.04

Number: 5000157180 Product: Z80/NSC800PASCAL 64823 01.02

One-line description:
Assignment to multi-dimensional array causes error 1006.

Problem:
Assignment to multi-dimensional arrays causes error 1006 to be generated. The following code is an example:

```
"BZ80"
PROGRAM TEST;
TYPE
  TEST_TYPE = ARRAY[1..2] OF CHAR;
  TABLE_TEST = ARRAY[1..2,1..2] OF TEST_TYPE;
VAR
  MTABLE : TABLE_TYPE;
  DUMMY : TEST_TYPE;

BEGIN
   DUMMY := MTABLE[1,1]       (*This causes Error 1006*)
END
.
```

Temporary solution:
Define the array as shown below:

```
"BZ80"
PROGRAM TEST;
TYPE
  TEST_TYPE = ARRAY[1..2] OF CHAR;
  TABLE_TWO = ARRAY[1..2] OF TEST_TYPE;
  TABLE_TYPE = ARRAY[1..2] OF TABLE_TWO;

VAR
  MTABLE : TABLE_TYPE;
  DUMMY1 : TABLE_TWO;
  DUMMY2 : TEST_TYPE;

BEGIN
  DUMMY1 := MTABLE[1];
  DUMMY2 := DUMMY1[1];
END
.
```

Signed off 04/29/87 in release 301.04

Number: D200020099 Product: Z80/NSC800PASCAL 64823 01.01

One-line description:
Compiler does not generate cross reference table.

Temporary solution:
To generate a cross reference table simply edit the source file and introduce an error (syntax error will do). The error will cause the compiler to generate the cross reference table. Once the table has been generated simply edit the source file and remove the error.

Signed off 04/29/87 in release 301.04

Number: D200029744 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: POINTERS

One-line description:
Variables of type pointer may not be incremented correctly.

Problem:

```
"PROCESSOR"
TYPE
PTR = ^BYTE;
TX  = PTR;

VAR
RXOUT: TX;
TEMP1,TEMP2 : BYTE;

BEGIN
TEMP1 := RXOUT^;
     LD     HL,[RXOUT]
     LD     A,[HL]
     LD     [TEMP1], A   ;HERE, TEMP1 IS CORRECTLY LOADED WITH THE BYTE
                         ;THAT RXOUT IS POINTING TO

RXOUT := TX(SIGNED_16(RXOUT)+1); {INCREMENT RXOUT}
     LD     HL,[RXOUT]
     INC    HL
     LD     [RXOUT],HL ;RXOUT IS CORRECTLY INCREMENTED

TEMP2 := RXOUT^;  {TEMP2 SHOULD GET THE NEXT BYTE}
     LD     [TEMP2],A  ;SINCE A WAS NOT DISTURBED, THE COMPILER DOES
                       ;NOT REALIZE THAT THE POINTER WAS UPDATED.
```

Temporary solution:
Set $AMNESIA ON$ around the pointer referencing code.

Signed off 04/29/87 in release 301.04

Number: D200037507 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: PASS 2

One-line description:
REBOOT DURING PASS 2 - related to position of variable declarations.

Problem:
The 64000 will reboot during pass 2 when compiling files where
1) The 105th external variable is an array, and
2) An element of the 105th external variable is accessed in the 19th procedure or function in the file (external and locally defined procedures count in this total).

Temporary solution:
Change the order of the external variable declarations, or change the order of the procedure declarations.

Signed off 04/29/87 in release 301.04

Number: D200062976 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
Error #1009 using byte-sized ORG'ed variables in FOR loops

Problem:
Error #1009 is generated when byte sized ORG'ed variables are used in FOR loops. The following code illustrates the problem.

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
PROCEDURE ERR;
VAR
$ORG 5000$
   B1,B2,X1: BYTE;

BEGIN

   FOR X1 := B1 to B2 DO;     (*Pass 2 Error 1009 - No free registers*)
END;
.
```

Temporary solution:
The error does not occur if the FOR loop variable is word sized instead of byte sized. It will also go away if the ORG statement is removed.

Signed off 04/29/87 in release 301.04

Number: D200062984 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
32-bit unsigned divide and modulus may fail

Problem:

The result of an unsigned 32-bit division or modulus operation may be incorrect if the dividend and the destination are the same location. The problem is in the library routine Zdworddiv. The following code demonstrates the problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2 : UNSIGNED_32;
BEGIN
  B1 := UNSIGNED_32(0E00000000);
  B2 := UNSIGNED_32(0900000000);
  B1 := B1/B2;
END.
```

Signed off 04/29/87 in release 301.04

Number: D200062992 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
Library routine REAL_ROUND may fail.

Problem:
The library routine REAL_ROUND may fail, causing floating point numbers to be incorrectly rounded to integers.

Signed off 04/29/87 in release 301.04

Number: D200063008 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
Set comparisons with the empty set may fail

Problem:
Set comparisons with the empty set may fail. The following code is an example of this problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
TYPE
  CH = 0..127;
  SET1 = SET OF CH;
VAR
  S1 : SET1;
PROCEDURE ERROR; EXTERNAL;
BEGIN
  S1 := [];
  IF S1 <> [] THEN          (*In CONST_prog, not enough bytes are
    ERROR;                    defined for the set*)
END.
```

Signed off 04/29/87 in release 301.04

Number: D200063016 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
DEBUG byte division and modulus may incorrectly report division by zero

Problem:
The DEBUG library routines for performing signed and unsigned byte division and modulus operations may fail and incorrectly report an attempted division by zero.

The following code fails in this manner:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2,B3 : BYTE;
$ORG 5000H$
  BA : ARRAY[1..15] OF BYTE;

BEGIN
  B1 := 1;
  B2 := 1;
  B3 := 0;
  BA[B3] := B1 DIV B2;   (*DIV fails - reports division by zero*)
END.
```

Signed off 04/29/87 in release 301.04

Number: D200063214 Product: Z80/NSC800PASCAL 64823 01.03

Keywords: PASS 3

One-line description:
Error 1113 generated during pass 3 when 23rd label is encountered.

Problem:
The compiler generates an error 1113 when it encounters the 23rd LABEL statement in a source file. The output listing shows that the compiler actually generates an internal label twice in the same code segment, which confuses the program counter.

Temporary solution:
Remove this label and write code which achieves the same function without using a GOTO statement.

Signed off 04/29/87 in release 301.04

Number: D200065292 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
Assignment of constant string of length 1 to string variable may fail.

Problem:
Assignment of a constant string of length 1 to a string variable that is itself a multidimensional array element may fail.

First, the address of the destination string is calculated in HL. Then the value of the string length resulting from the assignment, i.e. one (1), is loaded into the position reserved for the length of the string via a store indirect through HL. Up to this point all is as it should be; however, the value of the single character that comprises the string is then also stored HL indirect, overwriting the length and failing to correctly load the string value. The HL register should be incremented before the second store.

The following is an example:

```
"processor name"
PROGRAM TEST;
TYPE
  STRING_15 = PACKED ARRAY[0..15] OF CHAR;
VAR
  TWO_D_ARR : ARRAY[1..3,1..3] OF STRING_15;
BEGIN
  TWO_D_ARR[2,1] := " ";
        LD   HL,0030H
        PUSH HL
        LD   HL,00002H
        PUSH HL
        LD   HL,00010H
        PUSH HL
        LD   HL,00001H
        PUSH HL
        LD   BC,DTEST-00040H
        LD   A,002H
        CALL Zarrayref
        LD   A,001H
        LD   [HL],A      (*or LD   M,A  *)
        LD   A,020H
        LD   [HL],A       (*This is the error - should INC HL first*)
END.
```

Signed off 04/29/87 in release 301.04

Number: 5000136986 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: ENHANCEMENT

One-line description:
More accurate error message when wrong parm type is passed to STRWRITE.

Problem:
If you pass STRWRITE a two dimensional array (which is illegal) it will generate error 1106.

```
"BZ80"

$EXTENSIONS ON$

PROGRAM STWRITE;

TYPE   STRING_4 = PACKED ARRAY[0..3] OF CHAR;
```



```
VAR    Y    :    ARRAY[0..3,0..3] OF STRING_4;
       Z    :    ARRAY[0..3] OF STRING_4;
       E    :    STRING_4;
       T    :    SIGNED_16;


BEGIN
   E := 'TOTO';
   STRWRITE(Y[1,1],1,T,E:4);
END.
```

Passing a two dimensional array (Y[] in this example) is incorrect because STRWRITE expects a 'string' type variable. A 'string' type variable is defined as a packed array of [0..n] (note: one dimension). It would be nice, however, if a meaningful error message was generated.

Signed off 04/29/87 in release 301.04

Number: D200051599 Product: Z80/NSC800PASCAL 300 64823S004 01.00

Keywords: IF

One-line description:
IF B1 <rel-op> 82 THEN B1 := B1 - 1; {DOESN'T WORK}

Problem:
```
VAR B1, B2 : BYTE;

BEGIN
IF B1 (>|<|=|<=|>=) B2 THEN
B1 := B1 - 1;   {THE REGISTER CONTAINING B1 IS DECREMENTED, THEN
                 OVERWRITTEN BEFORE IT IS SAVED IN MEMORY}
```

Temporary solution:
$AMNESIA +$

Signed off 04/29/87 in release 401.20

Number: D200051854 Product: Z80/NSC800PASCAL 300 64823S004 01.00

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for adding one char to another.

Problem:
```
VAR
SRC, DEST : CHAR;

BEGIN
DEST := DEST + SRC;  {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 04/29/87 in release 401.20

Number: D200064311 Product: Z80/NSC800PASCAL 300 64823S004 01.10

One-line description:
Error #1009 using byte-sized ORG'ed variables in FOR loops

Problem:
Error #1009 is generated when byte sized ORG'ed variables are used in FOR loops. The following code illustrates the problem.

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
PROCEDURE ERR;
VAR
$ORG 5000$
   B1,B2,X1: BYTE;

BEGIN
```



```
   FOR X1 := B1 to B2 DO;     (*Pass 2 Error 1009 - No free registers*)
END;
```

Temporary solution:
The error does not occur if the FOR loop variable is word sized instead of byte sized. It will also go away if the ORG statement is removed.

Signed off 04/29/87 in release 401.20

Number: D200064410 Product: Z80/NSC800PASCAL 300 64823S004 01.10

One-line description:
32-bit unsigned divide and modulus may fail

Problem:
The result of an unsigned 32-bit division or modulus operation may be incorrect if the dividend and the destination are the same location. The problem is in the library routine Zdworddiv. The following code demonstrates the problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2 : UNSIGNED_32;
BEGIN
  B1 := UNSIGNED_32(0E00000000);
  B2 := UNSIGNED_32(0900000000);
  B1 := B1/B2;
END.
```

Signed off 04/29/87 in release 401.20

Number: D200064485 Product: Z80/NSC800PASCAL 300 64823S004 01.10

One-line description:
Library routine REAL_ROUND may fail.

Problem:
The library routine REAL_ROUND may fail, causing floating point numbers to be incorrectly rounded to integers.

Signed off 04/29/87 in release 401.20

Number: D200064550 Product: Z80/NSC800PASCAL 300 64823S004 01.10

One-line description:
DEBUG byte division and modulus may incorrectly report division by zero

Problem:
The DEBUG library routines for performing signed and unsigned byte division and modulus operations may fail and incorrectly report an attempted division by zero.

The following code fails in this manner:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2,B3 : BYTE;
$ORG 5000H$
  BA : ARRAY[1..15] OF BYTE;

BEGIN
  B1 := 1;
  B2 := 1;
  B3 := 0;
  BA[B3] := B1 DIV B2;   (*DIV fails - reports division by zero*)
END.
```

Signed off 04/29/87 in release 401.20

Number: D200064949 Product: Z80/NSC800PASCAL 300 64823S004 01.10

One-line description:
Set comparisons with the empty set may fail

Problem:
Set comparisons with the empty set may fail. The following code is an example of this problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
TYPE
  CH = 0..127;
  SET1 = SET OF CH;
VAR
  S1 : SET1;
PROCEDURE ERROR; EXTERNAL;
BEGIN
  S1 := [];
  IF S1 <> [] THEN          (*In CONST_prog, not enough bytes are
    ERROR;                    defined for the set*)
END.
```

Signed off 04/29/87 in release 401.20

Number: D200065318 Product: Z80/NSC800PASCAL 300 64823S004 01.10

One-line description:
Assignment of constant string of length 1 to string variable may fail.

Signed off 04/29/87 in release 401.20

Number: D200029777 Product: Z80/NSC800PASCAL 500 64823S001 01.10

Keywords: POINTERS

One-line description:
Variables of type pointer may not be incremented correctly.

Problem:

```
"PROCESSOR"
TYPE
PTR = ^BYTE;
TX  = PTR;

VAR
RXOUT: TX;
TEMP1,TEMP2 : BYTE;

BEGIN
TEMP1 := RXOUT^;
     LD     HL,[RXOUT]
     LD     A,[HL]
     LD     [TEMP1], A   ;HERE, TEMP1 IS CORRECTLY LOADED WITH THE BYTE
                         ;THAT RXOUT IS POINTING TO

RXOUT := TX(SIGNED_16(RXOUT)+1); {INCREMENT RXOUT}
     LD     HL,[RXOUT]
     INC    HL
     LD     [RXOUT],HL ;RXOUT IS CORRECTLY INCREMENTED

TEMP2 := RXOUT^;  {TEMP2 SHOULD GET THE NEXT BYTE}
     LD     [TEMP2],A  ;SINCE A WAS NOT DISTURBED, THE COMPILER DOES
                       ;NOT REALIZE THAT THE POINTER WAS UPDATED.
```

Temporary solution:
Set $AMNESIA ON$ around the pointer referencing code.

Signed off 04/29/87 in release 101.50

Number: D200036673 Product: Z80/NSC800PASCAL 500 64823S001 01.20

Keywords: IF

One-line description:
IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}

Problem:

```
VAR B1, B2 : BYTE;

BEGIN
IF B1 (>|<|=|<=|>=) B2 THEN
B1 := B1 - 1;   {THE REGISTER CONTAINING B1 IS DECREMENTED, THEN
                 OVERWRITTEN BEFORE IT IS SAVED IN MEMORY}
```

Temporary solution:
$AMNESIA +$

Signed off 04/29/87 in release 101.50

Number: D200040105 Product: Z80/NSC800PASCAL 500 64823S001 01.20

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for adding one char to another.

Problem:

```
VAR
SRC, DEST : CHAR;

BEGIN
DEST := DEST + SRC;  {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 04/29/87 in release 101.50

Number: D200064295 Product: Z80/NSC800PASCAL 500 64823S001 01.40

One-line description:
Error #1009 using byte-sized ORG'ed variables in FOR loops

Problem:
Error #1009 is generated when byte sized ORG'ed variables are used in FOR loops. The following code illustrates the problem.

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
PROCEDURE ERR;
VAR
$ORG 5000$
   B1,B2,X1: BYTE;

BEGIN

   FOR X1 := B1 to B2 DO;     (*Pass 2 Error 1009 - No free registers*)
END;
.
```

Temporary solution:
The error does not occur if the FOR loop variable is word sized instead of byte sized. It will also go away if the ORG statement is removed.

Signed off 04/29/87 in release 101.50

Number: D200064394 Product: Z80/NSC800PASCAL 500 64823S001 01.40

One-line description:
32-bit unsigned divide and modulus may fail

Problem:
The result of an unsigned 32-bit division or modulus operation may

be incorrect if the dividend and the destination are the same location. The problem is in the library routine Zdworddiv. The following code demonstrates the problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2 : UNSIGNED_32;
BEGIN
  B1 := UNSIGNED_32(0E00000000);
  B2 := UNSIGNED_32(0900000000);
  B1 := B1/B2;
END.
```

Signed off 04/29/87 in release 101.50

Number: D200064469 Product: Z80/NSC800PASCAL 500 64823S001 01.40

One-line description:
Library routine REAL_ROUND may fail.

Problem:
The library routine REAL_ROUND may fail, causing floating point numbers to be incorrectly rounded to integers.

Signed off 04/29/87 in release 101.50

Number: D200064535 Product: Z80/NSC800PASCAL 500 64823S001 01.40

One-line description:
DEBUG byte division and modulus may incorrectly report division by zero

Problem:
The DEBUG library routines for performing signed and unsigned byte division and modulus operations may fail and incorrectly report an attempted division by zero.

The following code fails in this manner:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2,B3 : BYTE;
$ORG 5000H$
  BA : ARRAY[1..15] OF BYTE;

BEGIN
  B1 := 1;
  B2 := 1;
  B3 := 0;
  BA[B3] := B1 DIV B2;   (*DIV fails - reports division by zero*)
END.
```

Signed off 04/29/87 in release 101.50

Number: D200064923 Product: Z80/NSC800PASCAL 500 64823S001 01.40

One-line description:
Set comparisons with the empty set may fail

Problem:
Set comparisons with the empty set may fail. The following code is an example of this problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
TYPE
  CH = 0..127;
  SET1 = SET OF CH;
VAR
  S1 : SET1;
PROCEDURE ERROR; EXTERNAL;
BEGIN
  S1 := [];
  IF S1 <> [] THEN          (*In CONST_prog, not enough bytes are
    ERROR;                    defined for the set*)
END.
```

Signed off 04/29/87 in release 101.50

Number: D200065284 Product: Z80/NSC800PASCAL 500 64823S001 01.40

One-line description:
Assignment of constant string of length 1 to string variable may fail.

Problem:
Assignment of a constant string of length 1 to a string variable that is itself a multidimensional array element may fail.

First, the address of the destination string is calculated in HL. Then the value of the string length resulting from the assignment, i.e. one (1), is loaded into the position reserved for the length of the string via a store indirect through HL. Up to this point all is as it should be; however, the value of the single character that comprises the string is then also stored HL indirect, overwriting the length and failing to correctly load the string value. The HL register should be incremented before the second store.

The following is an example:

```
"processor name"
PROGRAM TEST;
TYPE
  STRING_15 = PACKED ARRAY[0..15] OF CHAR;
VAR
  TWO_D_ARR : ARRAY[1..3,1..3] OF STRING_15;
BEGIN
  TWO_D_ARR[2,1] := " ";
        LD   HL,0030H
        PUSH HL
        LD   HL,00002H
```



```
        PUSH HL
        LD   HL,00010H
        PUSH HL
        LD   HL,00001H
        PUSH HL
        LD   BC,DTEST-00040H
        LD   A,002H
        CALL Zarrayref
        LD   A,001H
        LD   [HL],A       (*or LD   M,A  *)
        LD   A,020H
        LD   [HL],A        (*This is the error - should INC HL first*)
END.
```

Signed off 04/29/87 in release 101.50

Number: D200029785 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

Keywords: POINTERS

One-line description:
Variables of type pointer may not be incremented correctly.

Problem:
```
"PROCESSOR"
TYPE
PTR = ^BYTE;
TX  = PTR;

VAR
RXOUT: TX;
TEMP1,TEMP2 : BYTE;

BEGIN
TEMP1 := RXOUT^;
     LD     HL,[RXOUT]
     LD     A,[HL]
     LD     [TEMP1], A    ;HERE, TEMP1 IS CORRECTLY LOADED WITH THE BYTE
                          ;THAT RXOUT IS POINTING TO

RXOUT := TX(SIGNED_16(RXOUT)+1); {INCREMENT RXOUT}
     LD     HL,[RXOUT]
     INC    HL
     LD     [RXOUT],HL ;RXOUT IS CORRECTLY INCREMENTED

TEMP2 := RXOUT^;  {TEMP2 SHOULD GET THE NEXT BYTE}
     LD     [TEMP2],A  ;SINCE A WAS NOT DISTURBED, THE COMPILER DOES
                       ;NOT REALIZE THAT THE POINTER WAS UPDATED.
```

Temporary solution:
Set $AMNESIA ON$ around the pointer referencing code.

Signed off 04/29/87 in release 301.70

Number: D200036681 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

Keywords: IF

One-line description:
IF B1 <rel-op> B2 THEN B1 := B1 - 1; {DOESN'T WORK}

Problem:
```
VAR B1, B2 : BYTE;

BEGIN
IF B1 (>|<|=|<=|>=) B2 THEN
B1 := B1 - 1;   {THE REGISTER CONTAINING B1 IS DECREMENTED, THEN
                 OVERWRITTEN BEFORE IT IS SAVED IN MEMORY}
```

Temporary solution:
$AMNESIA +$

Signed off 04/29/87 in release 301.70

Number: D200040113 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for adding one char to another.

Problem:
```
VAR
SRC, DEST : CHAR;

BEGIN
DEST := DEST + SRC;  {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 04/29/87 in release 301.70

Number: D200064303 Product: Z80/NSC800PASCAL VAX 64823S003 01.60

One-line description:
Error #1009 using byte-sized ORG'ed variables in FOR loops

Problem:
Error #1009 is generated when byte sized ORG'ed variables are used in FOR loops. The following code illustrates the problem.

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
PROCEDURE ERR;
VAR
$ORG 5000$
   B1,B2,X1: BYTE;

BEGIN

   FOR X1 := B1 to B2 DO;      (*Pass 2 Error 1009 - No free registers*)
END;
.
```

Temporary solution:
The error does not occur if the FOR loop variable is word sized instead of byte sized. It will also go away if the ORG statement is removed.

Signed off 04/29/87 in release 301.70

Number: D200064402 Product: Z80/NSC800PASCAL VAX 64823S003 01.60

One-line description:
32-bit unsigned divide and modulus may fail

Problem:
The result of an unsigned 32-bit division or modulus operation may

be incorrect if the dividend and the destination are the same location. The problem is in the library routine Zdworddiv. The following code demonstrates the problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2 : UNSIGNED_32;
BEGIN
  B1 := UNSIGNED_32(0E00000000);
  B2 := UNSIGNED_32(0900000000);
  B1 := B1/B2;
END.
```

Signed off 04/29/87 in release 301.70

Number: D200064477 Product: Z80/NSC800PASCAL VAX 64823S003 01.60

One-line description:
Library routine REAL_ROUND may fail.

Problem:
The library routine REAL_ROUND may fail, causing floating point numbers to be incorrectly rounded to integers.

Signed off 04/29/87 in release 301.70

Number: D200064543 Product: Z80/NSC800PASCAL VAX 64823S003 01.60

One-line description:
DEBUG byte division and modulus may incorrectly report division by zero

Problem:
The DEBUG library routines for performing signed and unsigned byte division and modulus operations may fail and incorrectly report an attempted division by zero.

The following code fails in this manner:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
VAR
  B1,B2,B3 : BYTE;
$ORG 5000H$
  BA : ARRAY[1..15] OF BYTE;

BEGIN
  B1 := 1;
  B2 := 1;
  B3 := 0;
  BA[B3] := B1 DIV B2;   (*DIV fails - reports division by zero*)
END.
```

Signed off 04/29/87 in release 301.70

Number: D200064931 Product: Z80/NSC800PASCAL VAX 64823S003 01.60

One-line description:
Set comparisons with the empty set may fail

Problem:
Set comparisons with the empty set may fail. The following code is an example of this problem:

```
"processor name"
PROGRAM TEST;
$EXTENSIONS ON$
TYPE
  CH = 0..127;
  SET1 = SET OF CH;
VAR
  S1 : SET1;
PROCEDURE ERROR; EXTERNAL;
BEGIN
  S1 := [];
  IF S1 <> [] THEN          (*In CONST_prog, not enough bytes are
    ERROR;                   defined for the set*)
END.
```

Signed off 04/29/87 in release 301.70

Number: D200065300 Product: Z80/NSC800PASCAL VAX 64823S003 01.60

One-line description:
Assignment of constant string of length 1 to string variable may fail.

Problem:
Assignment of a constant string of length 1 to a string variable that is itself a multidimensional array element may fail.

First, the address of the destination string is calculated in HL. Then the value of the string length resulting from the assignment, i.e. one (1), is loaded into the position reserved for the length of the string via a store indirect through HL. Up to this point all is as it should be; however, the value of the single character that comprises the string is then also stored HL indirect, overwriting the length and failing to correctly load the string value. The HL register should be incremented before the second store.

The following is an example:

```
"processor name"
PROGRAM TEST;
TYPE
  STRING_15 = PACKED ARRAY[0..15] OF CHAR;
VAR
  TWO_D_ARR : ARRAY[1..3,1..3] OF STRING_15;
BEGIN
  TWO_D_ARR[2,1] := " ";
        LD   HL,0030H
        PUSH HL
        LD   HL,00002H
```

```
        PUSH HL
        LD   HL,00010H
        PUSH HL
        LD   HL,00001H
        PUSH HL
        LD   BC,DTEST-00040H
        LD   A,002H
        CALL Zarrayref
        LD   A,001H
        LD   [HL],A      (*or LD   M,A  *)
        LD   A,020H
        LD   [HL],A       (*This is the error - should INC HL first*)
END.
```

Signed off 04/29/87 in release 301.70

Number: D200010132 Product: Z8000 C 64820 00.56

Keywords: PASS 1

One-line description:
Unsigned integers treated as signed when subtracted from pointers

Problem:
When an unsigned short or integer is used as an offset to a pointer, the unsigned will be treated as a signed when doing pointer calculations. Offsets large enough to set the sign bit will be interpreted as a negative offset when the offset is subtracted from a pointer. The following code exibits the problem if offset is greater than 32767 dec.

```
unsigned offset;
struct { int a,b,c;
       } *ptr;
unsigned long x;

main ()
{
  x = ptr - offset;  /* The compiler will generate code negating   */
}                    /* offset for the "-" operation.              */
```

Temporary solution:
Cast the offset in the expression as the next larger integer.
ie. x = ptr - (unsigned long)offset;

Signed off 04/29/87 in release 001.06

Number: D200011403 Product: Z8000 C 64820 00.56

Keywords: PASS 1

One-line description:
Functions invoked via function pointers may JSR the wrong location.

Problem:
When the typedef statement is used to define pointers to functions, and this pointer type is used in a cast of a variable array to invoke code stored in that array, program execution may transfer to the wrong location. For example, in the following code the simple call to code_array fails while the call and assignment to p works correctly:

```
     typedef int(*PFI)();  /* PFI a pointer to int functions */
     int code_array[100];  /* array contains code */
     PFI p;                /* p a pointer of type PFI */

     pfibug()
     {
       (*((PFI) code_array))();  /* fails in JSR to code_array */
       (*(p=(PFI)code_array))(); /* assignment and JSR successful */
     }
```

Temporary solution:
Set up a dummy variable and perform an assignment to it when doing this type of operation.

Signed off 04/29/87 in release 001.06

Number: D200014498 Product: Z8000 C 64820 01.03

One-line description:
RANGE_ON

Problem:
With RANGE_ON, typecasting an integer and assigning it to an unsigned_integer produces a range error.

Signed off 04/29/87 in release 001.06

Number: D200040345 Product: Z8000 C 64820 01.03

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                    case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 001.06

Number: D200059741 Product: Z8000 C 64820 01.04

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 001.06

Number: D200063461 Product: Z8000 C 64820 01.05

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 001.06

Number: D200066043 Product: Z8000 C 64820 01.05

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
```

```
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 001.06

Number: D200051938 Product: Z8000 C 300 64820S004 01.00

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"processor name"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"processor name"

main(){
        int  c;
             switch(c){
                  case1:         break;
                  default:       switch(c){
                                    case 2: break;
                                 }
                                 break;
             }
}
```

Signed off 04/29/87 in release 401.20

Number: D200059774 Product: Z8000 C 300 64820S004 01.00

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"
```

```
main() {

int  i;
struct  undefined   a[10][20];

)
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 401.20

Number: D200063495 Product: Z8000 C 300 64820S004 01.10

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 401.20

Number: D200066076 Product: Z8000 C 300 64820S004 01.10

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:
"C"
"processor"

```
char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

"C"
"processor"

```
char  string[]  =  "do it this way";

main()
```



```
{
  int i;

  i = sizeof(string);
)
```

Signed off 04/29/87 in release 401.20

Number: D200066431 Product: Z8000 C 300 64820S004 01.10

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 401.20

Number: D200040352 Product: Z8000 C 500 64820S001 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 101.60

Number: D200059758 Product: Z8000 C 500 64820S001 01.40

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {
```



```
int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 101.60

Number: D200063479 Product: Z8000 C 500 64820S001 01.50

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 101.60

Number: D200066050 Product: Z8000 C 500 64820S001 01.50

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;
```

```
  i = sizeof(string);
}
```

Signed off 04/29/87 in release 101.60

Number: D200066423 Product: Z8000 C 500 64820S001 01.50

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 101.60

Number: 1650018804 Product: Z8000 C VAX 64820S003 01.80

One-line description:
No error message for unimplemented processor name.

Problem:
Specifying an unimplemented processor name in a C source file will cause the compiler to go from pass 1 into C Nocode without an error message. The listing file also does not report the error.

Signed off 04/29/87 in release 301.90

Number: D200040360 Product: Z8000 C VAX 64820S003 01.20

One-line description:
Nested switch statements may generate infinite loop

Problem:
If you have nested switch statements and do not terminate the inner switch's cases with breaks the compiler generates an infinite loop.

```
"C"
"68000"

main(){
     int c;
          switch(c) {
                    case 1:    break;
                    default:   switch(c){
                                    case 2: break;
                               }
                    /* A break is needed here because the break
                       above for 'case 2' generates a jump to
                       this location.  If a break is not placed
                       here it falls into the code for
                       evaluating 'case 1' above.  */
          }
```

Temporary solution:
Close default statement with a break.

```
"C"
"68000"

main(){
        int  c;
             switch(c){
                  case1:        break;
                  default:      switch(c){
                                   case 2: break;
                                }
                                break;
             }
}
```

Signed off 04/29/87 in release 301.90

Number: D200059766 Product: Z8000 C VAX 64820S003 01.50

One-line description:
Compiler is not flagging an undefined structure.

Problem:
The customer reports that the program listed below causes the compiler to hang. I could not duplicate this problem, but, the compiler incorrectly reported no errors.

```
"C"
"processor"

main() {

int  i;
struct  undefined   a[10][20];

}
```

The compiler should report that the type 'undefined' is undefined.

Temporary solution:
No temporary solution.

Signed off 04/29/87 in release 301.90

Number: D200063487 Product: Z8000 C VAX 64820S003 01.80

One-line description:
C Function returning large (>2bytes) result can't be called as procedure

Problem:
Functions returning large (>2byte) result cannot be called as procedures.

Signed off 04/29/87 in release 301.90

Number: D200066068 Product: Z8000 C VAX 64820S003 01.80

One-line description:
Illegal forward reference flagged for legally defined string.

Problem:

```
"C"
"processor"


char   badstring[] = {"Wont work"};
char   string[] = "works fine";

main()
{
  int i;

  i = sizeof(string);
```



```
  i = sizeof(badstring);        /* Error 117 flagged. */
}
```

Temporary solution:
Eliminate the braces when initializing a string.

```
"C"
"processor"

char  string[]  =  "do it this way";

main()
{
  int i;

  i = sizeof(string);
}
```

Signed off 04/29/87 in release 301.90

5958-6019, May 1987
E0587

Printed in U.S.A.